

Debian勉強会資料集

2005年冬号

あんどきゅめんとつど  
でびあん



東京エリアDebian勉強会 著

## 目次

1	ITP の仕方からアップロードまでの流れ	2
2	debconf 論	7
3	apt-listbugs の生い立ちと実装	17
4	“claim” makes Debian better	25
5	Starting debconf translation	29
6	debbugs internal	30
7	Debian とステータス	38
8	Debian Weekly News 翻訳フロー	42
9	Debian Weekly News trivia quiz	49
10	Debian Weekly News 問題回答	70

### 『あんどきゅめんでっど でびあん』について

本書は、東京周辺で毎月行なわれている『東京エリア Debian 勉強会』で使用された資料・小ネタ・必殺技などを一冊にまとめたものです。収録範囲は勉強会第7回から第10回まで。内容は無保証、つっこみなどがあれば勉強会にて。

## 1 ITP の仕方からアップロードまでの流れ

岩松 信洋



### 1.1 はじめに

今回、「このソフトウェアを Debian パッケージにして Debian の一部として提供したい」と思っているんだけどどうしたらいいのか、Debian Developer ではない人の視点から、ITP をどのように行えばいいのかまとめてみました。

### 1.2 ITP とは

Intend To Package の略で新しく Debian にパッケージを提供したいときに BTS に登録します。これは Debian Developer でなくても行うことができます。

### 1.3 ITP をする前に

ITP をする前にチェックしておかなければいけないことがあります。

#### 1. パッケージ化したいソフトウェアのライセンスをチェックします。

Debian は Debian フリーソフトウェアガイドライン ( DFSG ) に準じているソフトウェアでないといけません。"GPL"、"BSD"、および "Artistic" がそのライセンスの例になります。

#### 2. もうすでにパッケージがある可能性があるのでチェックする。

<http://www.debian.org/distrib/packages> や apt-cache search で確認することができます。

パッケージ化されていても Orphaned ( みなしご化 ) されていることもあります。

<http://www.debian.org/devel/wnpp/orphaned> で確認することができます。

ITA (Intent To Adopt ) をして引き取ってメンテナンスをしましょう。

#### 3. ITP されていないか確認する。

もうすでにパッケージにしたいソフトウェアが ITP されている可能性があります。

[http://www.debian.org/devel/wnpp/being\\_packaged](http://www.debian.org/devel/wnpp/being_packaged) で確認することができます。

#### 4. RFP されていることをチェックする。

Request For Package の略でパッケージ化の要望が BTS されていることがあります。

<http://www.debian.org/devel/wnpp/requested> で確認することができます。

RFP されているときはバグの内容を RFP から ITP に変更する必要があります。

#### 5. Debian パッケージにしてみる。

提供したいソフトウェアを Debian パッケージにしましょう。ITP してからパッケージ化してもいいと思いますが、先におこなっておいて不具合がないか調べておきましょう。

これらをチェックして ITP をしましょう。

## 1.4 ITP のしかた

実際にはどのように ITP をしたらよいのでしょうか。方法を以下にまとめてみました。

### 1.4.1 ITP

#### 1. reportbug の起動

```
% reportbug --email hoge@example.org wnpp
```

として reportbug を起動します。

--email で e-mail アドレスを指定します。

wnpp というのは Work-Needing and Prospective Packages の略です。

#### 2. バグリクエストタイプを聞かれるので ITP を選択します。

```
1 ITP This is an 'Intent To Package'. Please submit a package description
along with copyright and URL in such a report.
```

```
2 O The package has been 'Orphaned'. It needs a new maintainer as soon as
possible.
```

```
3 RFA This is a 'Request for Adoption'. Due to lack of time, resources,
interest or something similar, the current maintainer is asking for
someone else to maintain this package. He/she will maintain it in the
meantime, but perhaps not in the best possible way. In short: the
package needs a new maintainer.
```

```
4 RFH This is a 'Request For Help'. The current maintainer wants to continue
to maintain this package, but he/she needs some help to do this, because
his/her time is limited or the package is quite big and needs several
maintainers.
```

```
5 RFP This is a 'Request For Package'. You have found an interesting piece of
software and would like someone else to maintain it for Debian. Please
submit a package description along with copyright and URL in such a
report.
```

Choose the request type: 1

#### 3. パッケージ名を求められるので ITP したいパッケージ名を入力します。

```
Please enter the proposed package name: hoge
```

#### 4. パッケージの簡単な説明を求められるので入力します。

```
Checking status database...
```

```
Please briefly describe this package; this should be an appropriate short
description for the eventual package:
```

```
> hogehoge program
```

5. 以下のような画面になるので、ソフトウェアの内容 ( バージョン、提供元 Web サイト、ライセンス、説明文 ) を入力します。

```
Subject: ITP: hoge -- hogehoge program
Package: wnpp
Owner: hoge <hoge@example.org>
Severity: wishlist
```

\*\*\* Please type your report below this line \*\*\*

```
* Package name      : hoge
  Version           : x.y.z
  Upstream Author   : Name <somebody@example.org>
* URL               : http://www.example.org/
* License           : (GPL, LGPL, BSD, MIT/X, etc.)
  Description       : hogehoge program
```

(Include the long description here.)

```
-- System Information:
Debian Release: testing/unstable
APT prefers unstable
APT policy: (500, 'unstable')
Architecture: i386 (i686)
Shell: /bin/sh linked to /bin/bash
Kernel: Linux 2.6.12
Locale: LANG=ja_JP.eucJP, LC_CTYPE=ja_JP.eucJP (charmap=EUC-JP)
```

6. 入力が終わると送信確認が行われます。

Your report will be carbon-copied to debian-devel, per Debian policy.

Spawning sensible-editor...

No changes were made in the editor.

Report will be sent to "Debian Bug Tracking System" <submit@bugs.debian.org>

Submit this report on wnpp (e to edit) [y|n|a|c|E|i|l|m|p|q|?]?

”y” を押すことによってサーバーに送信されます。これで ITP の作業は終わりです。

#### 1.4.2 RFP ITP

パッケージ化を希望されているパッケージをパッケージ化し、Debian に提供する方法です。RFP のバグに対して BTS します。実際にはタイトルを RFP から ITP に変更します。

以下のようなメールをバグ番号に対して送信します。バグコントロールメールサーバにコマンドを送信しています。

```
owner 283119 ! /// address を #bugnumber の「所有者」に設定
retitle 283119 ITP: libflash -- GPL Flash (SWF) Library /// タイトルを
変更する
thanks /// コントロールサーバーへのコマンド終了
```

Hi,

I am interested in this package.  
I wish to adopt this package.

Thanks,  
Iwamatsu

#### 1.4.3 O ITA

みなしご化されているパッケージを引き取ってメンテナンスしたいときの方法です。Orphaned のバグに対してメールをします。実際にはタイトルを O から ITA に変更します。

以下のようなメールをバグ番号に対して送信します。バグコントロールメールサーバにコマンドを送信しています。

```
owner 283119 ! /// address を #bugnumber の「所有者」に設定
retitle 283119 ITA: libflash -- GPL Flash (SWF) Library /// タイトルを
変更する
thanks /// コントロールサーバーへのコマンド終了
```

Hi,

I am interested in this package.  
I wish to adopt this package.

Thanks,  
Iwamatsu

#### 1.5 ITP したら

1. スポンサーを探す ITP をしたら スポンサーを探す必要があります。スポンサーとは スポンサーは現存する Debian Developer で 自分の指導者をしてくれる方です。Debian Developer でないとパッケージを Debian にアップロードすることができません。スポンサーは作成したパッケージをチェックし、パッケージをアップロードしてくれます。また、間違いがあるときは指摘をしてくれたり、アドバイスをしてくれます。  
スポンサーを探すには debian-mentors ML で聞いてみるとよいでしょう。

<http://lists.debian.org/debian-mentors/>

また、GPG キーサインでサインして頂いた Debian Developer の方に相談するのも方法のひとつです。

2. スポンサーが見つかったらスポンサーが見つかったらそのスポンサーとやり取りし、パッケージを改善します。

スポンサーがアップロードしてもいいと判断された場合、スポンサーの手によって Debian にアップロードされます。

## 2 debconf 論

鵜飼文敏<sup>\*1</sup>



### 2.1 はじめに

debconf とは、Debian においてパッケージの設定を行なうためのフレームワークおよびそれを実装したパッケージです。そもそもは、元 Debian Project Leader の Wichert Akkerman 発案の configuration database framework 構想<sup>\*2</sup>であり、debconf は、それに基づく Joey Hess による実装です。

従来、パッケージの設定のうち、パッケージメンテナがデフォルトを決めにくいもの、つまりシステム管理者によって決定されるようなものは、メンテナスクリプト<sup>\*3</sup>で、プロンプトをだし管理者が入力した値に基づいて設定ファイルを生成するようにしていました。これはこれで非常に柔軟性が高い<sup>\*4</sup>のですが、パッケージによって<sup>\*5</sup> やりかたが異なってしまう、ディストリビューションとしての統一感に欠けてしまうという欠点がありました。また、パッケージのインストール時・アップグレード時に常に管理者の介在が必要になってしまうために、インストール・アップグレード中にコンソールを離れているわけにはいかないという問題もはらんでいます。

そこで考案されたのが debconf です。debconf を使うことで設定データを統一して扱うことができるようになります。

### 2.2 debconf を使っているパッケージの設定の仕方

apt-utils をインストールしておくこと apt-extracttemplates を使って、パッケージをダウンロードしてインストール作業をする前<sup>\*6</sup>に debconf を使ってパッケージの設定をおこなうことができます。この時は debconf 自体の設定 debconf/frontend に従ったフロントエンド<sup>\*7</sup>をつかってユーザとのやりとりがおこなわれます。また、設定には優先度 (priority) <sup>\*8</sup>がつけられており、debconf/priority の値より優先度が高いものだけがフロントエンドを介してユーザとやりとりをおこなわれるようになります。

またインストールした後も dpkg-reconfigure でパッケージの設定をしなおすことができます。dpkg-reconfigure は、debconf/priority の値にかかわらず低優先度 (low) <sup>\*9</sup>以上すなわちすべての設定をやりなおすようになっています。

また、現在そのパッケージの設定値がどうなっているかを知りたい場合は debconf-show コマンドが使えます。

---

<sup>\*1</sup> Fumitoshi UKAI, ukai@debian.or.jp, ukai@debian.org, Debian Project

<sup>\*2</sup> /usr/share/doc/debian-policy/debconf.specification.html

<sup>\*3</sup> postinst など

<sup>\*4</sup> メンテナががんばってメンテナスクリプトを書けばいろいろできる

<sup>\*5</sup> メンテナによってというべきか?

<sup>\*6</sup> unpack をする前

<sup>\*7</sup> dialog, readline, gnome, kde, editor, noninteractive のどれか

<sup>\*8</sup> 重要 (critical)、高 (high)、中 (medium)、低 (low) のどれか

<sup>\*9</sup> -p オプションを使わなければ

```
# debconf-show debconf
* debconf/priority: critical
* debconf/frontend: Dialog
```

## 2.3 debconf の裏側

apt-get がパッケージをダウンロードしてくると/etc/apt/apt.conf.d/70debconf というファイルにより dpkg-preconfigure -apt が実行されるようになります。

```
// Pre-configure all packages with debconf before they are installed.
// If you don't like it, comment it out.
DPkg::Pre-Install-Pkgs {"usr/sbin/dpkg-preconfigure --apt || true"};
```

dpkg-preconfigure では、apt-utils の apt-extracttemplates を使っているため、apt-utils がインストールされていないと、debconf の設定はこのタイミングではおこなわれません。

dpkg-preconfigure -apt に対して、いまダウンロードしたパッケージのリストがわたされるので、それらから template をとりだして config スクリプトの実行をおこないます。

- apt-get による \*.deb のダウンロード。 /var/cache/apt/archives/ にパッケージがおかれる
- dpkg-preconfigure -apt の実行
  - /var/cache/apt/archives/ からインストールしようとする deb をみつける
  - control 情報の templates と config をとりだす<sup>\*10</sup>
  - templates を load して、debconf データベース<sup>\*11</sup>を更新
  - config スクリプトを実行する。db\_set、db\_input や db\_go
- dpkg -unpack
  - preinst を実行
  - パッケージを展開し、ファイルをおきかえ
- dpkg -configure
  - postinst を実行。db\_get

基本的には、config スクリプトで設定情報を決定し<sup>\*12</sup>、postinst スクリプトでそれを読みだして実際のパッケージの設定ファイルなどに書きだすという処理をおこなうことになります。

なお、debconf データベースは registry として使う<sup>\*13</sup>ので、debconf データベースを参照するのはメンテナスクリプトだけにしておく必要があります。また、debconf 以外で設定ファイルを変更された場合も、その情報を上書きすることなく反映する必要があります。

debconf では、メンテナスクリプトと debconf データベース間のやりとりのプロトコルを決めています。こうすることで、フロントエンドをかえたりバックエンドのデータベースの実装を変更したりすることができるようになっているわけです。

<sup>\*10</sup> dpkg -I パッケージ.deb templates config。apt-extractpackage を使う

<sup>\*11</sup> /var/cache/debconf/\*.data

<sup>\*12</sup> ユーザに対して設定情報をたずねるか、デフォルト値を設定する

<sup>\*13</sup> 使うと debconf abuse として bug をくらう

シェルコマンド	内容	データのむき	使う場所
db_version "2.0"	バージョンネゴシエーション		
db_capb multiselect	キャパビリティネゴシエーション		
db_title タイトル	タイトル文字列の設定	スクリプト ユーザ	config
db_stop	debconf の停止		postinst, postrm
db_input プライオリティ 変数名	変数への入力	テンプレ ユーザ DB	config
db_go	入力の実行		config
db_get 変数名	変数のとりだし	DB スクリプト	postinst
db_set 変数名 値	変数の設定	スクリプト DB	config
db_reset 変数名	変数の初期化	テンプレ DB	
db_subst 変数名 鍵 置換	置き換え	テンプレ ( スクリプト)	config
db_fget 変数名 フラグ	フラグのとりだし	DB スクリプト	
db_fset 変数名 フラグ 値	フラグの設定	スクリプト DB	
db_metaget 変数名 フィールド	フィールドのとりだし	テンプレ スクリプト	
db_register テンプレート 変数名	変数の生成	元テンプレ テンプレ	config
db_unregister 変数名	変数の削除	テンプレ	
db_purge	データベースから削除	テンプレ、DB	postrm

表 1 debconf コマンド

### 2.3.1 テンプレートと変数

debconf では templates ファイルでテンプレートを定義し、そのテンプレートで作られる変数に対して debconf プロトコルを使って値を設定したり、読みだしたりしています。テンプレートを定義するとそれと同名の変数がつくられるので、ほとんどの場合ではテンプレート名と同名の変数を使うことがほとんどですが、同じような情報をいくつか持ちたい場合などではテンプレートから複数の変数を生成する場合があります。

テンプレートは debian/templates ファイルもしくは debian/パッケージ名.templates ファイルに記述します。

```

Template: 名前
Type: 型
Default: デフォルト値
Description: 短かい説明
長い説明

```

名前としては基本的には「パッケージ名/識別子」という文字列を使います。パッケージ名がブレフィクスについているので、他のパッケージのテンプレートと衝突することはありません。もし複数のパッケージで共有するようなテンプレートは「share/共有名/識別子」のような名前をつかうことが推奨されています。

型としては次のようなものがあります。

debconf で使うメッセージは翻訳する場合、debconf-gettextize を使って debian/po/ディレクトリ以下に各国語版の po ファイルを置けるように準備をします。

```

% debconf-gettextize templates

To complete conversion, you must:
a. Check that new templates files contain all the localized fields
b. Add po-debconf to Build-Depends or Build-Depends-Indep in debian/control
c. Remove obsolete files:
   templates.old
d. Edit debian/rules to generate the localized templates file, either
   with dh_installdebconf or directly with po2debconf

```

型名	意味
string	文字列
boolean	true か false
select	Choices:に設定されている値から一つ (“,” で区切る)
multiselect	select と違い複数選べる
note	Description:の内容を提示するだけ。メールとしても送られる
text	Description:の内容を提示する
password	パスワード用。

表2 テンプレートの Type

このように templates ファイルを指示して debconf-gettextize を実行すると、元の templates ファイルは .old サフィックスがついたものに残され、新しい templates ファイルが作られます。新しい templates ファイルは、翻訳すべきフィールドは “\_” がプレフィクスとしてつくようになります。そして po ディレクトリに POTFILES.in と templates.pot が生成されます。翻訳する場合は、templates.pot をロケール名.po にコピーして gettext を翻訳するのと同じようにして翻訳していきます。マルチパッケージで複数の templates ファイルがある場合は、それをすべて debconf-gettextize 実行時に指示します。古い templates.old は消してしまってもかまいません。

このようにして作られる翻訳を deb パッケージにちゃんととりこめるようにするには、まず debian/control の Build-Depends(か Build-Depends-Indep) に po-debconf を追加します。

後は debian/rules の binary-arch、binary-indep ルールで dh\_installdebconf を dh\_installdeb の直前くらいに呼ぶようにしておけば後は dh\_installdebconf がしかるべき処理をしてくれます。cdbs を使っている場合は include /usr/share/cdb/1/rules/debhelper.mk すれば中で dh\_installdebconf を実行してくれます。また Depends: 行に \$misc:Depends をいれるのを忘れないようにしましょう。dh\_installdebconf がしかるべき依存関係を設定してくれます。

L10N バグ (po 翻訳) を受けとった時は、そのファイルを debian/po ディレクトリにしかるべき名前 (ロケール名.po) で置くだけで OK です。

### 2.3.2 config スクリプト

config スクリプトは、deb パッケージがインストールされる前に<sup>\*14</sup>実行されます。config スクリプトでやるべきことは基本的には以下のような処理になります。

```
#!/bin/sh -e
# sample config
#
. /usr/share/debconf/confmodule
db_version 2.0
db_capb multiselect
if [ -f /etc/default/mypackage ]; then
. /etc/default/mypackage
db_set mypackage/foo "$FOOR"
db_set mypackage/bar "$BAR"
db_go
fi

db_title "My Package Configuration"
db_input low mypackage/foo || true
db_input low mypackage/bar || true
db_go
```

config スクリプトで使う debconf コマンドは以下のとおり

- . /usr/share/debconf/confmodule

<sup>\*14</sup> 実際には postinst などから confmodule が読みこまれた時にも

まず /usr/share/debconf/confmodule を . でよみこみます。これで frontend と backend のプロセスにわかれそれぞれ通信するようになります。ちなみに frontend は perl で書かれていて open2 でスクリプトを起動しなおして通信しています。なお debconf シェルモジュールではすべてのコマンドには db\_ が前についてるが、プロトコル上はこの db\_ は実際にはつきません。

- (必要があれば)db\_version でバージョンチェック。  
通信に使う debconf プロトコルのバージョンをネゴシエーションします。これは「version 2.0 で通信したい」といっているわけで、それが debconf でサポートできないようなバージョンだとリターンコード 30 を変数 \$RET にいれてかえします。この場合エラーになるので sh -e によりここで実行が終了します。このコマンドは postinst なんかでも使います。
- (必要があれば)db\_capb で capability チェック  
必要とするキャパビリティを要求します。指定できるのは backup および multiselect のふたつ。backup は前の質問に戻るをサポート、multiselect は複数のセレクトをできるようにする機能です。
- 設定ファイル<sup>\*15</sup>があるかどうかをチェックして、設定ファイルで指定されている現在の設定をよみとります。  
もしあれば、その値で db\_set して、debconf 変数の値に反映させます。

```
db_set 変数名 値
```

db\_set はユーザとのやりとりは発生しません。

- db\_title でタイトルの設定  
質問ダイアログを出すときのタイトルを設定します。
- db\_input でユーザに聞く質問を指示

```
db_input プライオリティ 変数名 || true
```

もしプライオリティが debconf 設定のプライオリティより高ければ変数名に対応しているテンプレートを使ってユーザにデータを入力させるようにする。つまり db\_input low だとその質問はあまり重要ではなく細かい設定がしたい人だけがやればよいというような項目になります。逆に db\_input critical だとユーザが与えないとまともな設定ができないような項目になります。

プライオリティ	意味
low	ほとんどのケースでデフォルト値で問題ない場合
medium	まともなデフォルト値がある場合
high	まともなデフォルト値がない場合
critical	デフォルト値では問題があるような場合。ユーザの入力が必要

表 3 プライオリティ

どのようにデータを入力するかはそのテンプレートにわりあてられている型によってかわります。

もしユーザに提示しなかったらエラーで \$RET が 30 になります。そのため通常は —— true として sh -e によりここでスクリプトが終了しないようにします。

なお、db\_input だけでは入力の指示をフロントエンドに送るだけで、質問自体はまだおこな

\*15 通常/etc/default/パッケージ

われません。実際にフロントエンドがユーザにたずねるのは `db_go` を呼び出した時です。

- `db_go`

`db_input` で与えられてきた「質問してよー」というのを実際にユーザに提示します。ここでダイアログがでてきてユーザとやりとりすることになります。

`debconf` は一度、`db_input` で入力した場合、その変数の `seen` フラグを `true` に設定します。このフラグの値を変更するには `db_fset` を使います。

```
db_fset 変数名 seen false
```

デフォルト値に戻したい場合は `db_reset` を使います。これでテンプレートに記述されていたデフォルト値に戻すことができます。

```
db_reset 変数名
```

基本的にはテンプレートと同名の変数を使うことで事足りる場合が多いですが、必要があればテンプレートから新しい変数を作ったりすることができます。変数をつくったりけしたりする操作は `db_register`、`db_unregister` を使います。

```
db_register テンプレート 変数名
```

```
db_unregister 変数名
```

また、変数を新たにつくった場合は質問のメッセージの一部を変えることが多いでしょう\*<sup>16</sup>。そのために `db_subst` というのが使えます。テンプレートの中で `${ 鍵文字列 }` というのを埋めこんでいて、次のように `db_subst` を実行すると、`${ 鍵文字列 }` の部分が、“置換文字列” に置きかえられます。

```
Template: mypackage/baz
...
Description: ${鍵文字列} の値?
${鍵文字列}の値を入力してください
```

```
db_register mypackage/baz mypackage/baz2
db_subst 変数名 鍵文字列 置換文字列
```

```
...
Description: 置換文字列 の値?
置換文字列の値を入力してください
```

### 2.3.3 postinst スクリプト

`postinst` スクリプト\*<sup>17</sup>では `debconf` からデータをよみだして実際の設定ファイルを生成することが仕事となります。`debconf` 以前では `postinst` 自身が `echo` や `read` コマンドなどをつかってユーザの入力をいれていたのをここでは `debconf` からとってくるようにするわけです。

\*<sup>16</sup> でないと、どれがどれかユーザにわからなくなってしまう

\*<sup>17</sup> `preinst` スクリプトはパッケージ展開前なので普通は `debconf` を使うことはない

```

#!/bin/sh -e
# postinst
. /usr/share/debconf/confmodule

case "$1" in
configure)
  db_get mypackage/foo
  FOO="$RET"
  db_get mypackage/bar
  BAR="$RET"
  if [ -f /etc/default/mypackage ]; then
    sed -e 's/^FOO=.*FOO="\"$FOO\""/' \
        -e 's/^BAR=.*BAR="\"$BAR\""/' \
        < /etc/default/mypackage > /etc/default/mypackage.dpkg-tmp
    if cmp -s /etc/default/mypackage /etc/default/mypackage.dpkg-tmp; then
      rm -f /etc/default/mypackage.dpkg-tmp
    else
      mv -f /etc/default/mypackage /etc/default/mypackage.dpkg-old
      mv /etc/default/mypackage.dpkg-tmp /etc/default/mypackage
    fi
  else
    cat <<DEFAULT > /etc/default/mypackage
# mypackage configuration file
# see /usr/share/doc/mypackage/README.Debian.
# this file is automatically managed by debconf.
#
# FOO="..."
# foo is blah blah
FOO="$FOO"
#
# BAR="..."
# bar is blah blha
BAR="$BAR"
# END OF FILE
DEFAULT
fi
;;
abort-upgrade|abort-remove|abort-deconfigure)
;;
*)
  echo "postinst called with unknown argument '$1'" >&2
  exit 1
;;
esac
db_stop
#DEBHELPER#
exit 0

```

config スクリプトで使う debconf コマンドは以下のとおり

- `. /usr/share/debconf/confmodule`  
 config スクリプトと同様、まず `/usr/share/debconf/confmodule` を `.` でよみこみます。注意すべきことは、ここで `config` スクリプトもまた実行されるということです。

- `db_get` で値のとりだし

db\_get 変数名

`db_get` を実行すると、変数名であらわされる変数に格納されている値を `$RET` にとりだすことができます。

- `db_stop` で debconf の終了

ここで debconf の処理を終了させます。これ移行、標準入出力が使えるようになります。たとえば `invoke-rc.d` など起動されるものがある場合、メッセージを標準出力にだそうとするのでこの前に `db_stop` しておかなければいけません。

`postinst` では、この例にあるように既存の設定ファイルをできるだけ維持しつつ、更新された設定値だけをいれかえるようにすることが期待されています。

### 2.3.4 postrm スクリプト

`postrm` スクリプトでは、このパッケージの debconf データを debconf データベースから削除しておく必要があります。

```
#!/bin/sh -e
#
case "$1" in
purse
. /usr/share/debconf/confmodule
db_purge
db_stop
;;
remove|upgrade|failed-upgrade|abort-install|abort-upgrade|disapper)
;;
*)
echo "$0 called with unknown argument '$1'" >&2
exit 0
;;
esac
#DEBHELPER#
```

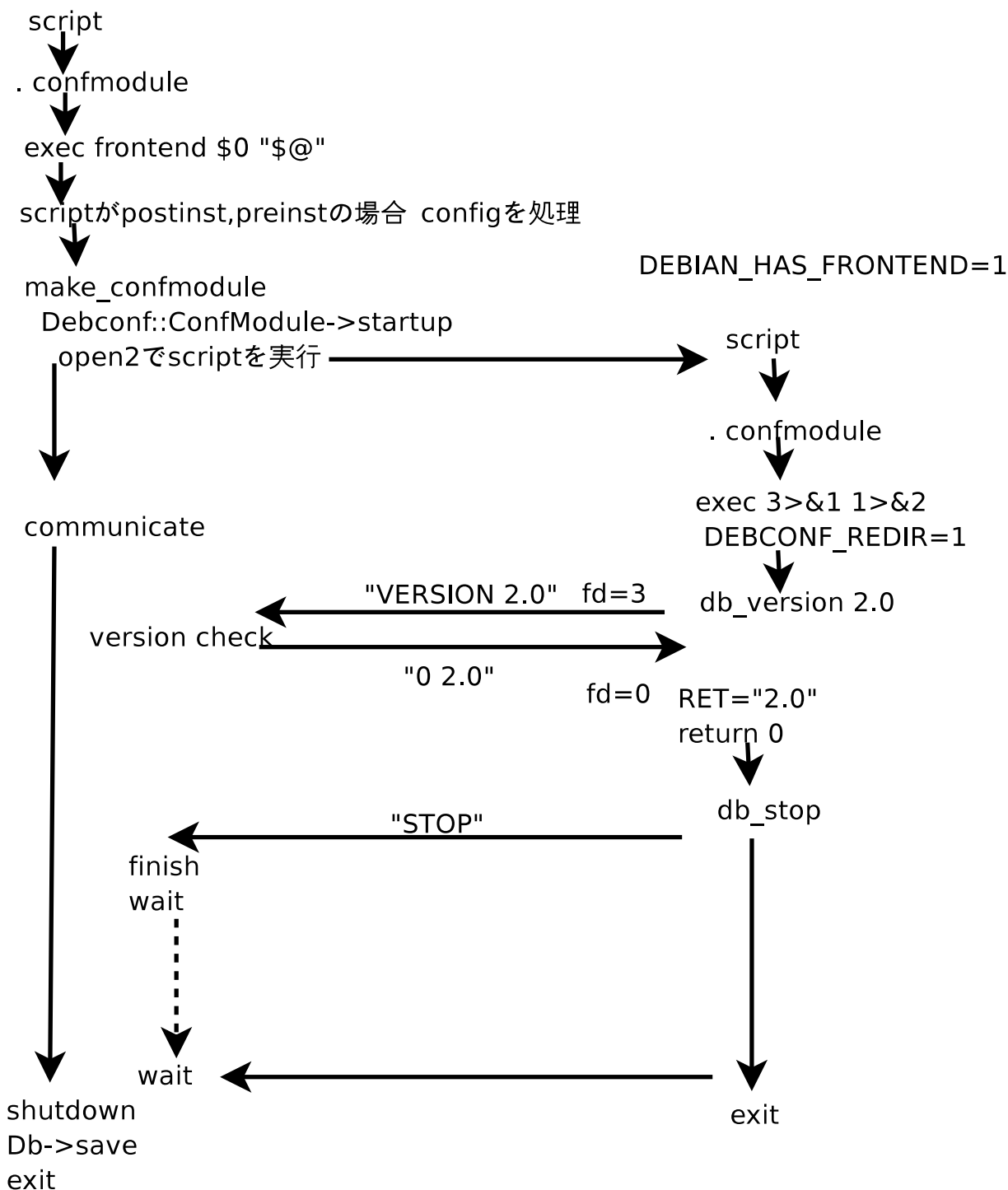
- . /usr/share/debconf/confmodule  
debconf を使う前にまず /usr/share/debconf/confmodule を . でよみこみます。
- db\_purge  
db\_purge でこのパッケージに属する debconf データを debconf データベースから削除します。
- db\_stop  
もう debconf は必要ないので stop します。

### 2.3.5 debconf プロトコルのやりとり

debconf は confmodule をソースすることで、frontend に exec しています。その中で confmodule を呼びだした script を open2 で起動して db\_コマンドを frontend との通信にして処理をおこなっています。

fd=3 にコマンドを出力すると、frontend プログラムで解釈実行され結果がかえってきます。その結果はスクリプトの標準入力に与えられるので read で読みとってスペースの前がコマンドの終了コード、スペースの後が\$RET に格納される値となります。

スクリプトがおわると frontend はそれを検知してデータベースに書き出して終了となります。



## 2.4 debconf データベース

debconf で設定したデータはどこにあるのでしょうか？ この設定は `/etc/debconf.conf` に記述されています。

デフォルトでは `/var/cache/debconf` に次のようなファイルとして格納されています。

debconf データベースの内容は `debconf-get-selections` を使うとダンプすることができます。この出力を `debconf-set-selections` の入力として渡すとロードすることができます。<sup>\*18</sup>

<sup>\*18</sup> `debconf-get-selections` は TAB で分割していて、`debconf-set-selections` では (特にタイプと値の間は) 空白文字列一つだけという点に注意したほうがいいでしょう。ファイル経由でやりとりしたりパイプ経由の場合は問題ありませんが、ターミナルの出力をコピーするとは

ファイル名	内容
templates.dat	templates の情報
config.dat	設定データ
password.dat	パスワードデータ

表 4 debconf データベース

debconf-get-selections -installer で、debian-installer で使って debconf データベースをとりだすことができます。

その他に debconf-copydb を使うことでデータベースファイルをコピーすることができます。

## 2.5 debconf を使っているスクリプトのデバッグ

debconf を使ってるスクリプトのデバッグは簡単ではありません。しかし基本は DEBCONF\_DEBUG に値を設定してスクリプトを実行すればいい場合が多いでしょう。

```
# DEBCONF_DEBUG='.*' /var/lib/dpkg/info/パッケージ.postinst configure 最後に設定されたバージョン
```

これでパッケージの設定する時の debconf の動きを追うことができます。なお簡単にするために dpkg-reconfigure debconf で debconf フロントエンドを Readline などをしておいた方がいいでしょう。

DEBCONF\_DEBUG='.\*' というのは DEBCONF\_DEBUG=user、DEBCONF\_DEBUG=developer、DEBCONF\_DEBUG=db すべてを指定したのと同じ意味になります。

debconf-communicate を使うと、debconf データベースと直接やりとりすることができます。標準入力からコマンドを与えると、標準出力に結果をかえします。コマンドは debconf シェルモジュールでつかうコマンドから db\_ をとりのぞいたものになることに注意しましょう。例えば次のように使います。

```
# echo 'get debconf/priority' | debconf-communicate
0 critical
```

0 が成功を意味<sup>\*19</sup>し、critical が debconf/priority の値<sup>\*20</sup>を意味します。

## 2.6 おわりに

本文書では Debian で設定データの管理を司っている debconf について解説しました。

まることがあります (TAB が複数のスペースになってしまっている)

\*19 db\_get の終了値

\*20 db\_get を実行した場合だと \$RET に設定される値

## 3 apt-listbugs の生い立ちと実装

樽石



この文書は 2005 年 3 月に北京で開催された Asia Debian Mini-Conf in Beijing で発表した資料の日本語訳です。

### 3.1 はじめに

Debian は 24 時間 365 日、多くの開発者によって日夜開発されている自由なオペレーティングシステムです。わたしたちはこのような最新のスナップショットを Debian のアップグレードフレームワークを使うことで簡単に共有することができ、実際に、このフレームワークを利用して多くの人が最新の Debian スナップショットを利用しています。

一方で、この最新スナップショットはときどき壊れることがあります。apt-listbugs は、このような壊れたスナップショットからあなたのコンピュータを守るための仕組みです。

この短い文書では、apt-listbugs がどのようにコンピュータを守るのか、apt-listbugs の生い立ち、実装について簡単に説明します。また apt-listbugs の抱えている現在の問題点について紹介します。

### 3.2 仕組み

Debian には Debian バグ追跡システム (BTS) として知られる、Debian に関する全ての問題が利用可能な中央管理型バグ追跡システムがあります。Debian のパッケージインストールフレームワークである APT は apt-get や aptitude 等を利用してインストールやアップグレードを行う直前に BTS からこれらの問題を取得するために apt-listbugs を呼び出します。apt-listbugs はどのパッケージがインストールされるのか、またアップグレードされるのかを自動認識し、問題となるバグを見付けた場合はその事をユーザに通知し、インストールを継続するか警告を出します。

このアプローチは、主にふたつの問題、ひとつはユーザの視点、もうひとつは開発者の視点を解決します。

#### 3.2.1 ユーザの視点

ユーザの視点は apt-listbugs のメインの目的です。APT フレームワークは任意のパッケージを簡単にアップグレードすることができます。複雑な作業を一切行わずに単に 'apt-get upgrade' と実行するだけです。これはすばらしい機能ですが、一方でこのアプローチは壊れたパッケージさえも簡単にインストールさせてしまいます。多くの人が apt-get upgrade を利用しているため、結果として世界中の Debian マシンが一瞬のうちに壊れてしまうこととなります。

実際は、壊れたパッケージの情報は BTS に既に存在している可能性が非常に高いのですがこの情報が BTS にあるかどうかを毎回チェックしないとイケないとしたら apt-get upgrade による簡単なアップグレード作業は面倒な作業に一変してしまいます。これが apt-listbugs が必要な理由です。apt-listbugs はそのような情報が BTS にあるかどうかをあなたの代わりに毎回自動で行い

ます。

### 3.2.2 開発者の視点

ふたつ目の目的は Debian 開発者のためのものです。基本的には、BTS は Debian パッケージに関するあらゆる情報知るために利用できる素晴らしいものです。もし自分のパッケージにバグがあった場合、他のユーザが問題を BTS に報告してくれます。開発者はいつでも、どこでも、この情報にアクセスすることができます。

しかし、仮に自分の環境ではたまたま発生しなかった致命的な問題が最新のパッケージに含まれてしまったとします。こうなると、多くのユーザは問題の深刻さを考慮して急いでバグ報告をしようとしてしまいます。それゆえ、たったひとつのバグに対して非常にたくさんのバグ報告を受け取ることになってしまい、開発者はバグ報告の分類という本質的でない作業に時間をとられ、本当に直したい致命的な問題の解決に長い時間を要してしまうこととなります。もし、ユーザがバグをレポートする前に似たようなレポートを閲覧することができれば、重複したレポートを送ることはなくなるでしょう。

## 3.3 ヒストリ

apt-listbugs はもともとユーザの視点をなんとかしたいという個人的な目的で数年前に開発されました。ちょうどその時は、大学院を修了するために修士論文を書かなければいけない時期でした。ところが、何を思ったか切の一週間前に apt-get upgrade をしてしまったのです。結果、NFS が動かなくなり、この問題を修正しなければいけなくなりました。このバグレポートをきちんとチェックしていれば、このような問題にはあわなかったのですが、バグレポートを毎回チェックするわけにはいきませんでした。なぜならいつインストールされるかわからない致命的な問題のために毎回バグレポートをチェックするのは非常に大変な作業だからです。

### 3.3.1 CGI 問題

apt-listbugs の最初のバージョンは Debian にすぐにアップロードすることはできませんでした。その理由は apt-listbugs は BTS の CGI 出力を利用していたからです。CGI スクリプトはスクリプト言語で書かれていたため、もしこのパッケージをアップロードすると、BTS サーバが非常に高負荷になってしまうことは容易に想像できます。そこで、この apt-listbugs を利用したら同時にアクセスできるユーザ数は何人程度なのか知るために CGI のパフォーマンスを測定することにしました。結果は、1 パッケージのレポートを取得するのにかかる時間は約 1 秒でした。BTS サーバは 2 台の CPU を備えていましたので、結局 1 秒に 2 つのバグレポートしか取得できません。そのため、もし 10 個のパッケージをアップグレードしようとしたら、レポートの取得に 5 秒かかってしまいます。これは非常に大きな問題です。特に、BTS サーバは Debian のマスタサーバでしたから、もし Debian ユーザ全員がそのサーバからバグレポートを取得したら、これはほとんど DoS アタックのような状態に陥ってしまいます。

### 3.3.2 強制キャッシュ

最初に考えた解決法は CGI 出力のキャッシュを行うプロキシサーバを利用することでした。プロキシサーバが http の No-Cache 属性を無視すれば、静的なデータを利用することができます。もちろん TTL の問題はありますが、DoS になってしまうよりは良いでしょう。

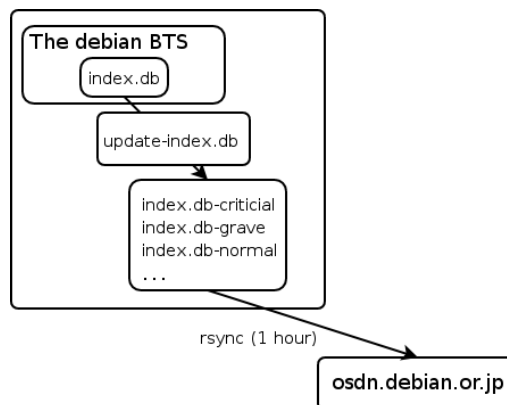
このサーバを用意したことで、とりあえず Debian に apt-listbugs をアップロードすることが

できましたが、はじめは experimental にアップロードして様子を見ていました。

### 3.3.3 index.db パーサ

はじめのバージョンの apt-listbugs はパーサに潜在的な問題を抱えていました。それはパーサが CGI 出力を利用していたことです。CGI のフォーマットはドキュメント化されていません。また、CGI スクリプト自体遅いという問題もあります。そのため、別の新しいパーサが必要でした。これは index.db パーサと呼ばれるパーサです。index.db パーサは BTS の内部データベースである index.db ファイルを直接パースするパーサです。index.db ファイル自体もドキュメント化はされていないのですが、このファイルは静的に生成されるものですので、CGI よりはずっと良い解です。

実際には、このパーサは index.db ファイルを critical や grave といった重要度毎に分割した index.db ファイルを利用します。



apt-listbugs には LDAP プロトコルを利用した実験的な LDAP パーサも存在しますが、このパーサは CGI パーサの代わりにするには意味がないものでした。その理由は当時の BTS LDAP サーバは内部的に tcl スクリプトを毎回よんでいるものだったからです。速度に変化はありませんでした。

Andreas Barth 氏は BTS 用のネイティブな LDAP インタフェースを作成しました。そのため、現在ではこのインタフェースを利用してバグを取得することが可能になっています。詳細は <http://people.debian.org/~aba/bts2/ldap/> を参照してください。ただ、このインタフェースを利用するインタフェースはまだ実装されていません。

### 3.3.4 SpamAssasin と apt-listbugs

Index.db パーサを利用することで、apt-listbugs は動的なデータをサーバから取得することはなくなりました。そこで、この時点で一時的な解決であった強制キャッシュサーバを停止しました。

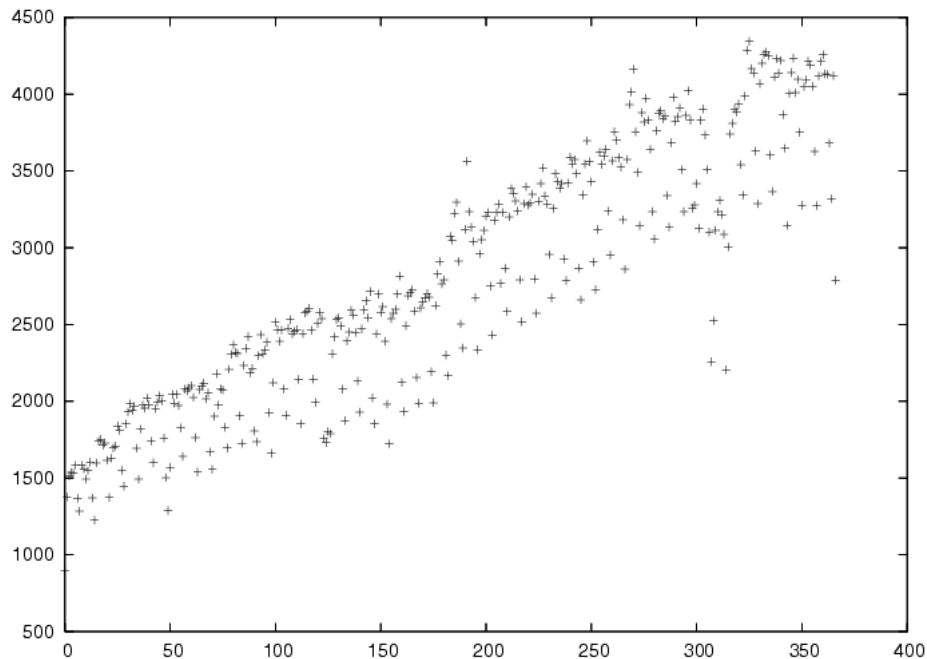
しかし、これは別の問題を生み出したのです。ちょうどその頃、BTS に対して非常に多くの SPAM メールが送られるようになっており、spamassasin が BTS サーバの CPU を非常に多く使うようになっていたのです。そして CPU 大量消費の原因が apt-listbugs にあるという勘違いされてしまったのです (Bug#207415)。

よく考えてみると、1 日で 46,000 個の静的データが負荷をこれほどあげるはずはありません。秒に換算すればわずか 0.5 個のだからです。通常の Web サイトはもっと多くの要求を処理できています。データサイズに関してはどうでしょうか。ひとつのデータは約 100 バイトですから、こ

れも 50B/s 程度です。いずれにしてもこの問題の対策として apt-listbugs はサーバへの直接アクセスを禁止されてしまいました。

とはいえ、この問題が apt-listbugs によるものであったとしてもなかったとしても apt-listbugs がマスタサーバに直接アクセスするのはあまり良い考えではありません。そこでこの問題をきっかけに osdn.debian.or.jp にミラーサーバを構築しました。

現在、1 日に 4500 システムが apt-listbugs を使用しています。以下のグラフはどれくらいの Debian システムが apt-listbugs を利用しているかを表しています。X 軸は osdn.debian.or.jp にミラーサイトを構築した日からの日数、Y 軸は 1 日に osdn.debian.or.jp にアクセスする IP アドレスの数です。



### 3.4 実装

apt-listbugs は オブジェクト指向スクリプト言語のひとつである Ruby で記述されています。主に以下のクラスから構成されています。

- \* Acquire
  - \* HTTP
  - \* File
- \* Parser
  - \* CGI
  - \* Index
  - \* DSA
  - \* ReleaseCritical
- \* Bug
  - \* Factory
    - \* PackageFactory
    - \* StatusFactory

- \* BugsFactory
- \* Viewer
- \* SimpleViewer
- \* RSSViewer

#### 3.4.1 Acquire

Acquire はデータを取得するクラスです。このクラスはサブクラスに対するキャッシュ機構を実装しています。実際の取得メソッド実装されておらず、サブクラスで実装します。例えば、HTTP サブクラスはデータを http から取得し、File サブクラスはデータをファイルシステムから取得します。

#### 3.4.2 Parser

Parser クラスは Acquire クラスで取得されたデータをパースし、バグ情報を保持する Bugs クラスを生成します。例えば以下の ruby コードは CGI 出力から apt-listbugs のバグ情を取得します。

```
acq = Debian::BTS::Acquire::HTTP.new
cgi_parser = Debian::BTS::Parser::CGI.new(acq)
bugs = cgi_parser.parse("apt-listbugs")
bugs.each do |bug|
  p bug
end
```

##### - CGI

CGI パーサは bugs.debian.org から取得した CGI 出力をパースします。

##### - Index

Index パーサは BTS システムの生データである index.db ファイルをパースします。apt-listbugs のデフォルトパーサです。

##### - DSA

DSA パーサは Debian セキュリティアドバイザリのページをパースします。

##### - ReleaseCritical

ReleaseCritical は Release-Critical ページをパースします。

#### 3.4.3 Bug

Bug クラスは ID、題名、重要度、タグ等のバグ情報を保持します。

これらのクラスは汎用的に作成されています。そのため、apt-listbugs 以外のプログラムからでもこれらのクラスを利用することが可能です。

#### 3.4.4 Factory

Factory クラスは「仕様書」からインスタンスを生成します。実際の作成はサブクラスにより行われます。このクラスは進捗表示等の Factory に汎用的なメソッドのみを実装しています。

##### - PackageFactory

PackageFactory は .deb ファイルの一覧からパッケージ情報を生成します。

```
# creating new packages database
new_pkgs = Factory::PackageFactory.create(pkgnames) do |msg, val|
  config.frontend.progress(msg, val) if config.quiet == false
end
Factory::PackageFactory.delete_ignore_pkgs(new_pkgs)
```

この処理が行われている間は、apt-listbugs から以下のメッセージが出力されます。

パッケージフィールドを読み込んでいます...

- StatusFactory

StatusFactory は指定したパッケージ名に対するの現在のシステムの情報を取得します。この処理中は以下のメッセージが出力されます。

パッケージ状態を読み込んでいます...

- BugsFactory

BugsFactory は Bug クラスで表現されるバグ情報を生成します。基本的に、この Factory はパーサクラスの単なるラッパーですが、パーサクラスは一度にひとつのパッケージのみを処理するのに対し、BugsFactory は複数のパッケージを処理できます。

バグレポートを取得しています...

### 3.4.5 Viewer

Viewer クラスはバグを表示するビューアを提供します。

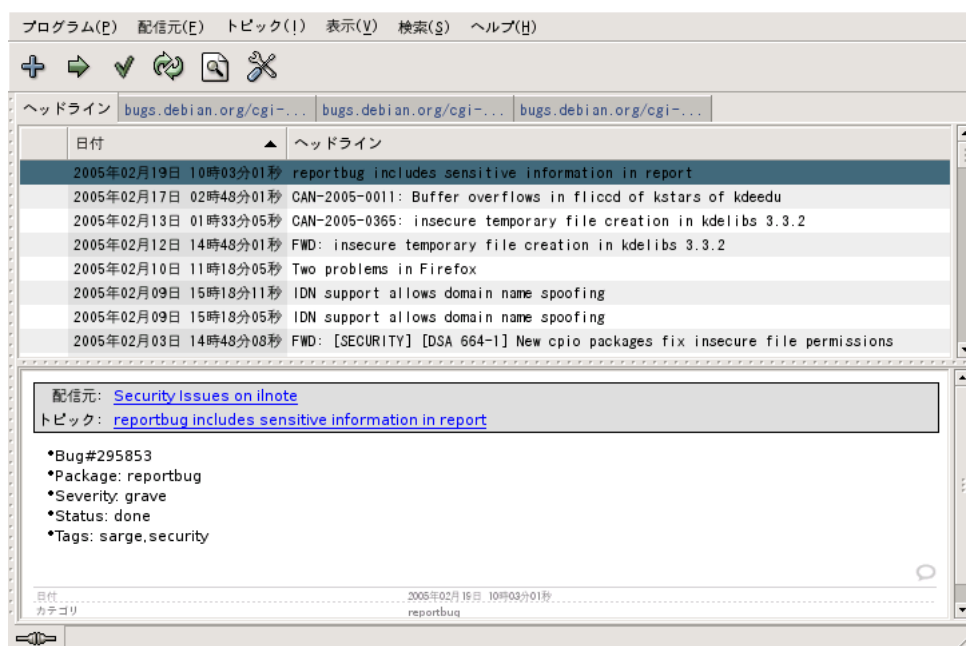
- SimpleViewer

SimpleViewer は以下のような処理を行う組み込みの対話的ビューアです。

```
critical bugs of cron (3.0pl1-86 -> 3.0pl1-87) <done>
#282722 - Network install of Debian Woody Alpha - cron corrupt on debian servers ?
grave bugs of strace (4.5.8-1.2 -> 4.5.9-1) <done>
#294172 - strace - builds no s390 binary
grave bugs of gaim (1:1.1.2-3 -> 1:1.1.3-1) <done>
#295904 - gaim: 1.1.3-1: dies with SIGABRT on startup
grave bugs of reportbug (3.7.1 -> 3.8) <done>
#295853 - reportbug includes sensitive information in report
grave bugs of cdb4 (0.4.26-4 -> 0.4.27-1) <open>
#295884 - cdb4: Changes control file to add new build dependency.
grave bugs of libdb4.3 (4.3.27-1 -> 4.3.27-2) <open>
#294163 - libdb4.3: build failed on hppa
Summary:
strace(1 bug), gaim(1 bug), cron(1 bug), cdb4(1 bug), libdb4.3(1 bug), reportbug(1 bug)
Are you sure you want to install/upgrade the above packages? [Y/n/?/...]
```

- RSSViewer

RSSViewer はバグ情報を RSS (Really Simple Syndication) 形式で出力します。



### 3.5 問題

現在、apt-listbugs が抱える一番の問題点は、apt-get upgrade を行うたびに apt-listbugs が大量のバグを表示してしまうことです。しかもほとんど役に立たない多くのレポートを読まなければいけません。それはそれほど重要でもない問題を重要だとタグ付けされた報告が非常に多いことが原因です。apt-listbugs は changelog ファイル中の "closes: Bug#XXXX" という形式の文字列を処理することで不要なレポートを表示しないようにしていますが、上記のようなレポートはパッケージ保守担当者によって手動でクローズされてしまうため、表示から取り除くことはできません。

バグには open, done 等の状態があるのですが、これを使うことはできません。それはバグの状態というのは終局的なシステムだからです。バグ報告は通常そのバグを修正した新しいパッケージが Debian にアップロードされるとすぐにクローズされます。この段階では、バグ状態は変更されますが、そのバグを修正したパッケージはまだ利用できません。ミラーサイト使っている場合、この遅延時間ももっと長くなるでしょう。

最初のバージョンの apt-listbugs はレポート文書中に存在する "Version" タグを利用しており、これによりいくつかのバグをフィルタすることができていました。この情報を利用すると

- 既にバグが存在している
- インストールしようとしているバージョンとは関係がない

といったフィルタリングが可能でした。しかし、現在は apt-listbugs が CGI にアクセスできないため、この情報を利用することができません。おそらく Since や Fixed-In といったような汎用的なタグ情報が必要であると考えています。

### 3.6 Tips

apt-listbugs はいろいろな利用方法があるためここでいくつか紹介します。

### 3.6.1 RSS によるセキュリティ問題

```
dpkg --get-selections | grep -v deinstall | awk -F' ' '{print $1;}' | \
  xargs -n $num /usr/sbin/apt-listbugs -y -q -T security --title \
  "Security Issues on 'hostname --fqdn'" rss
```

このスクリプトは `/usr/share/doc/apt-listbugs/security` にあります。

### 3.6.2 作業中のパッケージで解決していないバグを見る

```
grep -e ^Package: < debian/control | cut -d' ' -f2- | \
  xargs /usr/sbin/apt-listbugs -S outstanding,open \
  critical,grave,serious,important,normal,wishlist,minor list
```

このスクリプトは `/usr/share/doc/apt-listbugs/deblistbugs` にあります。

## 3.7 まとめ

apt-listbugs はまだいろいろ問題を抱えていますが、基本的な概念は非常におもしろいものです。現在の興味は apt-listbugs に webrick 等を使って組み込み http サーバを導入することです。このフロントエンドを利用すると apt-listbugs の管理が非常に便利になります。また、bts2ldap を利用した LDAP バックエンドも必要であると考えています。

## 3.8 参考

- Debian - <http://debian.org/>
- Debian Bug Tracking System - <http://bugs.debian.org/>
- <http://lists.debian.org/debian-devel/2003/08/msg02376.html>

## 4 “claim” makes Debian better

やまね



### 4.1 本日の目的

Debian BTS について理解を深める

- BTSって何さ？
- どういうときにするの？
- 何がいいの？
- 実際どうすればいいの？

そして立派なクレーマーとして認められる！

### 4.2 Debian Bug Tracking System

Debian 独自のバグ追跡システム。システムとしては debbugs という独自のものを利用

特徴

- Web から閲覧可能（まあ、最近のは皆そうですね）
- やり取りは基本的に全てオープン
- メールベースで作業が進む（ここは珍しいかも）
- かなり使い込まれてます。30 万件近くが登録済み。  
redhat の bugzilla はこの半分ぐらいの件数

#### 4.2.1 どんなときに BTS を利用しますか？

- パッケージングのバグに遭遇したとき
- アップグレードしたらよくわからない現象が起こるようになってしまったとき
- いつまで経っても security fix が提供されないとき
- 気の利いた機能を実装したのでパッチを取り込んでもらいたいとき
- 地味～な L10N な作業を取り込んでもらうとき
- セキュリティホールの報告があったのでメンテナをせっつきたいとき

#### 4.2.2 擬似パッケージ (pseudo package)

パッケージではないが、BTS で扱うためにパッケージとして扱うもの

- Web サイト
- wnpp – 作業が望まれるパッケージ (ITP も)
- インストールシステム などなど



図 1 BTS のページ (<http://www.debian.org/Bugs/>)

<http://www.debian.org/Bugs/pseudo-packages.ja.html> 参照

ここでのポイント

あらゆる苦情・提案は BTS に集まる。誰も聞いていないところで文句を言うのではなく BTS すべし。

### 4.3 BTS 用ツール

#### 4.3.1 reportbug/querybts

reportbug コマンド

簡単にバグレポート・レポートの検索が可能

- 対話的な操作が可能です。
- レポートはメールで飛びます。ポーンと。
- レポートは gnupg で署名も可能。まるでちゃんとした報告みたいに見えます。

querybts コマンド

バグレポートの検索に特化しています。

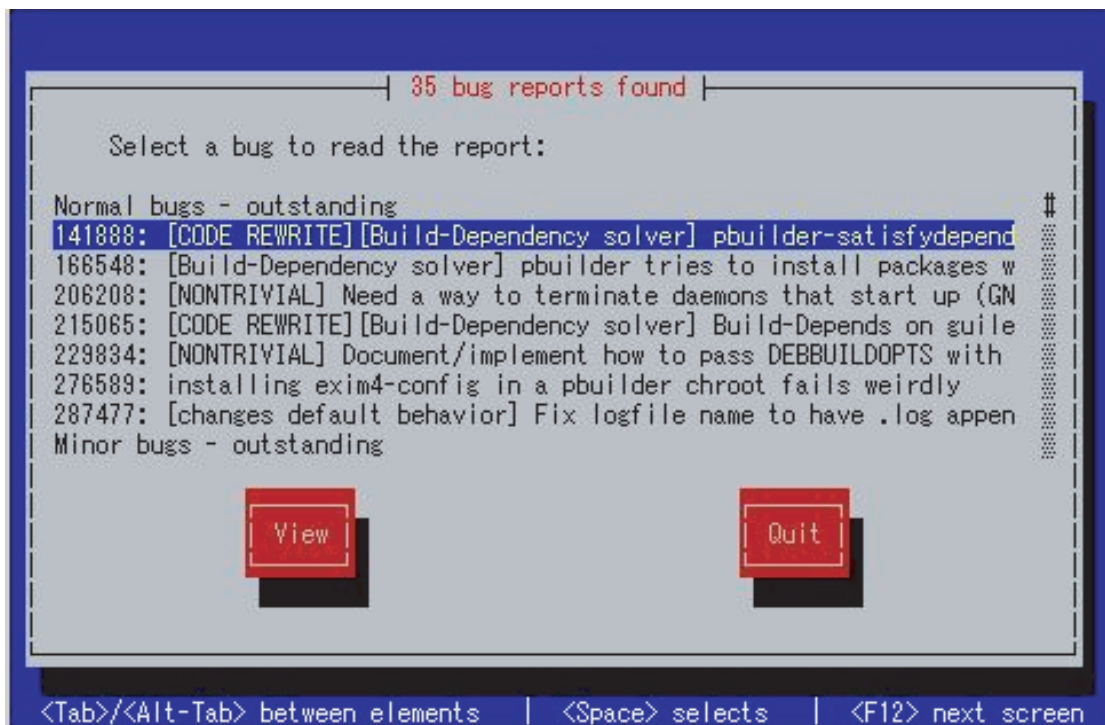


図 2 reportbug の画面

#### 4.3.2 debbugs-el

emacs ユーザは、debbugs-el パッケージに含まれる debian-bug コマンドを使うこともできます。M-x debian-bug と入力すると、reportbug とよく似たやり方で、全ての必要事項を尋ねられます...らしい。(emacs 使ってないので不明)

### 4.4 「重要度」と「タグ」

#### 4.4.1 Severity (重要度) レベル

<http://www.debian.org/Bugs/Developer#severities> 参照

- critical(致命的)
 

システム上の関係のないソフトウェア (またはシステム全体) を破壊する、重大なデータの欠落を引き起こす、または、そのパッケージをインストールしたシステム上でセキュリティホールが生じる場合。
- grave(重大)
 

問題のあるパッケージが使用できない、またはほとんど使用できない。またはデータの欠落を引き起こす、そのパッケージを使用するユーザのアカウントにアクセスを許してしまうセキュリティホールが生じる場合。
- serious(深刻)
 

Debian ポリシーに対して見すごせない違反がある (大まかに言うと、"must" や "required" の要件に違反している)、またはパッケージメンテナの意見としてそのパッケージがリリースに適していないと判断された場合。
- important(重要)

バグがパッケージの利用に大きく影響しており、対処しなければ誰にもまったく使用できない場合。

- normal(通常)  
デフォルト値。通常のバグ。
- minor(軽度)  
問題がパッケージの利用に影響しない、かつ修正はたいした事がないと思われる場合。
- wishlist(要望)  
将来的な要望、主に設計上の理由により修正が非常に困難なバグ。

#### 4.4.2 タグ

<http://www.debian.org/Bugs/Developer#tags> 参照

- patch(パッチ)  
バグ報告に、バグを修正するためのパッチや簡単な手順が含まれています。パッチがあってもバグを適切に解決できない場合や別の問題を生じる場合は、このタグは使うべきではありません。
- security(セキュリティ)  
このバグはパッケージのセキュリティ問題を説明します。ほとんどのセキュリティバグは、critical (致命的) や grave (重大) の severity (重要度) も設定すべきです。
- upstream(上流)  
このバグは、パッケージの上流の部分に影響します。
- d-i(インストーラ)  
このバグは、debian-installer に関するものです。インストーラの開発に関係するけれども、インストーラの直接の構成要素ではないパッケージに対するバグの場合、このタグを使ってください。
- L10n  
このバグは、パッケージの地域化に関するものです。
- woody / sarge / sid / experimental  
このバグは特に各ディストリビューション に加えられるものです。

#### 4.5 BTS の心得

- 何よりも相手を尊重しよう
- 事実を端的に述べよう
  - 環境の記述は多すぎず少なすぎずを目指そう
  - バージョンやアーキテクチャぐらい書こう (面倒な人はツールを使いましょう)
- broken English でもいいや、と開き直ろう
- でも多少は体裁は整えておこう
  - gnupg 使ってみるとか
  - 定型シグネチャ使っておくとか

## 5 Starting debconf translation

やまね



<http://kmuto.jp/debian/po-trans/> に翻訳の状況が掲載されている。

### 5.1 必要なもの

- debconf-updatepo  
po が最新のものをチェック
- msgfmt  
文法があっているかをチェック

### 5.2 手順

#### 1. po-debconf をインストール

```
apt-get install po-debconf
```

#### 2. 翻訳したいパッケージのソースをインストール

```
apt-get source <packagename>
```

#### 3. template.po をコピーして翻訳する

```
cd <packagename>/debian/po  
cp template.pot ja.po
```

#### 4. 査読してもらう

debian-doc@debian.or.jp とかに投げる

#### 5. BTS する

```
reportbug -A 翻訳したファイル -g  
ファイルを添付して GPG Sign してバグレポート送信
```

## 6 debugs internal

上川



### 6.1 はじめに

この文書は Anthony Towns がフィンランドの debconf 5 で発表した内容を日本にて展開するための資料です。Anthony Towns の作成した英語の資料を省略して抜粋しています。また、それ以降に変更した事項について追記しています。

Debian Bug Tracking System (BTS) は、ほぼ Debian に特化したバグ報告の管理のためのシステムです。他のプロジェクトでも利用されていることもありますが、Debian でバグがパッケージベースで厳格に分類できることなどの特性が反映されているため、Debian プロジェクトのワークフローで使いやすいように作られています。<sup>\*21</sup>

規模としては、55000 以上の現在アクティブなバグ報告、231000 のアーカイブされたバグ報告を現在保持していて、毎週 1000 以上の新規のバグ報告が追加されています。ウェブインタフェースは追加された報告をすぐに反映しており、過去、ダウンタイムもほとんど発生していません。

Anthony Towns によると下記がバグトラッキングシステムの要件です。

- インタフェース：開発者がメールで操作できるようになっており、誰でもウェブで閲覧できるようになっている。
- パッケージベース：バグ報告をパッケージ別に高速に管理する必要がある
- スケーラビリティ：大量のバグ報告に対応できる必要がある
- 即時性：現在のバグの状態をすぐに報告してくれる必要があり、バグの状態が変更されたらすぐに反映される必要がある
- 安定性：継続して動作する必要がある。新規の機能がどんどん追加されたとしても。
- 公開：議論の内容に Debian コミュニティ全体として参加できるように、永続的な公開記録として保存される必要がある。

### 6.2 データ形式

バグデータベースのスプールの形式は下記です。リレーショナルデータベースなどは利用していません、スプールディレクトリ以下にほとんどのデータが格納されています。

各バグについて、ファイルはそれぞれ 4 個あります。サマリーファイルはメタデータを保存します。ログファイルは、そのバグに対して流れたメールを全て保存します。

status ファイルは互換性のためだけに存在しています。report ファイルは、最初のバグ報告のメールで、バグが close されるときに送信されるものです。

- /org/bugs.debian.org/spool

---

<sup>\*21</sup> Debian のインフラと統合されており、changelog にバグ番号を記述してパッケージをアップロードしたらバグが修正されたと記録されるようになっていたりします。

```

- incoming/
  * T.*
  * S[BMQFDU RC] *.*
  * R[BMQFDU RC] *.*
  * I[BMQFDU RC] *.*
  * G[BMQFDU RC] *.*
  * P[BMQFDU RC] *.*
- db-h/
  * 00/
  . ..
  . 314200.log
  . 314200.report
  . 314200.status
  . 314200.summary
  * ..
  * 99/
- archive/
  * 00/
  * ..
  * 99/
- index.db – index.db.realtime へのシンボリックリンク
- index.archive – index.archive.realtime へのシンボリックリンク
- nextnumber

```

### 6.2.1 incoming

incoming に来たメールは処理中、名前を変えます。

- T receive によってうけとられた
- S SPAM 確認待ち
- R SPAM 確認中
- I SPAM チェック通った
- G service か process スクリプトを通った
- P process 中

また、ファイル名の二つ目の文字はどこのメールアドレスにメールが送信されてきたものなのかということを示します。ファイル名ののこりは、バグ番号と、一意な ID です。一意な ID を決定するのに現在は時間とプロセス番号を利用しています。

- B: 通常のバグ報告。submit@ 1234@
- M: -maintonly メーリングリストに投げない
- Q: BTS に登録しない。-quiet
- F: アップストリームにフォワード -forwarded
- D: バグ終了 -done

- U: サブミッターにメール -submitter
- R: ユーザのリクエスト用インタフェース request@
- C: デベロッパーの制御用インタフェース control@

### 6.2.2 Status と Summary

status ファイルの中身は行ベースです。無い行については空行とみなします。このファイルは今後なくしていこうとしています。

- バグ報告者のメールアドレス
- 時間 (秒)
- サブジェクト
- 元のメールのメッセージ ID
- バグがアサインされているパッケージ
- タグ
- close した人のメールアドレス
- 上流のメールアドレスか URL(forward されたばあい)
- マージされているバグ番号
- severity

summary ファイルは RFC822 形式で、拡張可能になっています。現在 Format-Version: 2 と 3 の二つの形式があります。3 は、ヘッダについては RFC1522(MIME) のデコードされた形式になっています。

- Format-Version: このファイル形式のバージョン
- Submitter: バグ報告者のメールアドレス
- Date: 時間 (秒)
- Subject: サブジェクト
- Message-ID: 元のメールのメッセージ ID
- Package: バグがアサインされているパッケージ
- Tags: タグ
- Done: close した人のメールアドレス
- Forwarded-To: 上流のメールアドレスか URL(forward されたばあい)
- Merged-With: マージされているバグ番号
- Severity: severity
- Owner: バグの所有者

### 6.2.3 log ファイル

あらゆるメールが log ファイルには追記されていきます。また、メタデータも追記されていきます。残念ながら、メタデータは生の HTML で書かれており、またバージョンによって記述の仕方が変わっており、さらに悪いことに、古いバグの中にあるテキストは更新されていないため、機械的に処理することは難しくなっています。

また、コントロール情報は、行頭のエスケープコードにより切り替わります。メールの中にエスケープコードのような文字列が出て来たら、それは文字コード 030(8進数)の文字を追加してエ

スケープします。

詳細は `Debbugs::Log` を見てください。

- `kill-init`: まだ一行も処理していません
- `incoming-recv: 07`: あとに `go` がくる、`Received`:行
- `autocheck: 01: X-Debian-Bugs-...`: までの無視されている行、`autowait` が次に来る
- `html: 06`: 生で表示すべき HTML
- `recips: 02`: メールの受取人、`04` で分割されている
- `go: 05`: メールの文書
- `go-nox: X`: メールの文書、`X` ではじまる行
- `kill-end: 03`: メッセージの終り。
- `autowait: go-nox` があとにくる、空行まで無視されるその他の情報。

#### 6.2.4 Index ファイル

`index` ファイルは、`pkgreport.cgi` がどのパッケージにどのバグがわりあてられているかを確認するための情報です。

以前は、`by-package.idx` と `by-severity.idx` というのがあり、高速化に貢献するはずだったのですが、一年以上長い間生成されていなかったうえに、生成されていなかったことに誰も気づかなかったので必要ないんじゃないだろうか、ということです。

データ形式としては下記のようになります。パッケージ、バグ番号、時間、ステータス、メールアドレス、`severity` の順に書いた行が全てのバグに対して作成されています。

```
pbuilder 317998 1121196782 open [Junichi Uekawa <dancer@netfort.gr.jp>] normal
```

### 6.3 コード形式

`debbugs` は特に設計もされずに長い間パッチを累積してきました。ただ、明確にわかれている部分はあって、メールを処理するコアのインタフェースのスクリプトと、ウェブを表示するための CGI 部分とで分離できます。

設定ファイルは全て `/etc/debbugs` にあります。

#### 6.3.1 コアのスクリプト

メールを処理する部分があります。

- `errorlib`: ライブラリ
- `receive`: MTA からメールを受信する
- `spamscan`: 受信メールを SPAM チェックする
- `processall`: `process` と `service` にメールを分配する
- `process`: バグメールを処理する
- `service`: `control@` と `report@` メールを処理
- `expire`: `close` されてから 28 日過ぎたバグをエキスパイア処理する
- `rebuild`: `index` ファイルをリビルド

`receive` と `rebuild` 以外は `cron` から起動しています。15 分に一回しか動作しません。

### 6.3.2 CGI スクリプト

CGI 関連は、errorlib 関数を活用している部分もありますが、ほぼ独立しています。

- bugreport.cgi: バグレポートを一つ表示
- pkgreport.cgi: パッケージやサブミッタなどでサマリを作成する
- pkgindex.cgi: パッケージや severity に対して数を表示
- common.pl: ライブラリとして利用

pkgreport.cgi はユーザが直接ウェブでたたくため、特に速度が重要視される部分なので、触る場合には注意してください。

### 6.3.3 ハックするには

debbugs のソースは CVS にあります。また、Debian Developer であれば、ミラーが merkel.debian.org の/org/bugs.debian.org 以下にあります。

## 6.4 そして何がおきたか

Anthony Towns の発表でどういう結果がもたらされたか見てみましょう。

### 6.4.1 バージョントラッキング

バグがどのバージョンで発見され、どのバージョンで修正されたのかというのをトラッキングできるようになりました。従来は発見されたバージョンだけが Version ヘッダで分かるようになったのですが、それ以外の情報も保持するようになりました。

<http://lists.debian.org/debian-devel-announce/2005/07/msg00010.html>

バグ番号を保持して操作するための BTS のコマンドは下記です。

```
close バグ番号 バージョン
reassign バグ番号 パッケージ バージョン
found バグ番号 バージョン
```

また、katie が変更され、バグを close するメッセージには、下記のヘッダが付くようになりました。それを BTS が処理して close されたバージョンを把握できるようになりました。

```
Source-Version: バグ番号
```

CGI に対して、version を指定すると、そのバージョンでの状態がでできます。329344<sup>\*22</sup>は、0.4 では open だったが、0.5 では close になったというのが <http://bugs.debian.org/cgi-bin/pkgreport.cgi?pkg=cowdancer&version=0.4> と <http://bugs.debian.org/cgi-bin/pkgreport.cgi?pkg=cowdancer&version=0.5> 二つのページを比較するとわかります。

</org/bugs.debian.org/spool/db-h/44/329344.summary> を見ると、メタデータとして保存されているのがわかります。

---

\*22 <http://bugs.debian.org/329344>

```
Format-Version: 2
Found-In: cowdancer/0.4
Done: Junichi Uekawa <dancer@debian.org>
Subject: cowdancer: cow-shell does not start, gives error
Date: 1127295198
Submitter: Francesco Potorti' <Potorti@isti.cnr.it>
Fixed-In: cowdancer/0.5
Package: cowdancer
Message-Id: <E1EI0YD-00031E-00@pot.isti.cnr.it>
Severity: grave
```

#### 6.4.2 ユーザタグ

<http://lists.debian.org/debian-devel-announce/2005/09/msg00002.html>

[request@bugs.debian.org](mailto:request@bugs.debian.org) に対して下記のようなメールをおくればタグが追加できます。

```
user aj@azure.humbug.org.au
usertag 18733 + good-reasons-to-run-for-dpl
usertag 18733 + still-cant-believe-it-finally-got-fixed
usertag 62529 + your-days-are-numbered
```

見る際には、users=でユーザを指定するとタグが見えるようになります。

<http://bugs.debian.org/cgi-bin/pkgreport.cgi?pkg=dlist;users=dancer@debian.org>

また、tag でタグを指定して、users=でユーザを指定するとそのユーザで作成したタグを全て検索することができます。

<http://bugs.debian.org/cgi-bin/pkgreport.cgi?tag=ignore-for-now;users=dancer@debian.org>

#### 6.4.3 バグ購読

バグ番号に対してメーリングリストのようにして利用することができるようになりました。

<http://lists.debian.org/debian-devel-announce/2005/07/msg00014.html>

バグ番号-subscribe@bugs.debian.org にメールを出すと登録するようにメールがかえってくるので、それに返信すると、バグ番号に登録されます。

#### 6.4.4 バグロッカー

どのバグがどのバグによって邪魔されているのかというのをトラッキングするための機能が追加されました。

```
block 保留中のバグ番号 by 原因のバグ番号
unblock 保留中のバグ番号 by 原因のバグ番号
```

#### 6.4.5 mindays maxdays

mindays, maxdays オプションが追加されました。バグ報告の報告されてからの日数で表示させるかさせないかを選択できるオプションです。

<http://bugs.debian.org/cgi-bin/pkgreport.cgi?maint=dancer@debian.org&maxdays=90> や <http://bugs.debian.org/cgi-bin/pkgreport.cgi?maint=dancer@debian.org&mindays=90> として入力できます。

#### 6.4.6 バグ検索システム

Google が master.debian.org にとって DoS になるような検索の仕方をしていたので、現在 BTS は google の検索対象にははっていないので検索サービスが必要だろう、という話題が出ていました。

鵜飼さんが全文検索エンジンサービス (FABRE) を実装しましたが、まだ本格的に使われるよ

うな状態にはまだいたっていないようです。結構このサービスは負荷が高いのが問題になると思われま。 <http://fabre.debian.net/>

#### 6.4.7 debian-bugs.el はまだ動くのか

reportbug は vi ユーザを中心としたインタフェースになっていますが、Emacs を利用している人でも、debian-bugs.el を利用して debian BTS を操作することができます。

たとえば、debian-changelog-mode を使っている場合であれば、changelog のエントリーを自動生成するようになっています。メニューから Bugs close を選択するか、debian-changelog-close-bug でバグ番号を選択する (タブ補完がききます) と、下記のようなエントリーが作成できます。

```
dsh (0.25.6-2) unstable; urgency=low

* Bug fix: "How to control dsh timeout time?", thanks to Junichi Uekawa
(Closes: #281012).
* Bug fix: "allow exclusion of host from list of hosts.", thanks to
Junichi Uekawa (Closes: #289766).
* Bug fix: "dsh: -c -i hangs if no input under current design", thanks
to Charles Fry (Closes: #241531).
```

この機能は 207852<sup>\*23</sup> で上川の出したパッチが発端で実装されましたが、気づいたら正規表現が巨大になっています。

ひさしぶりにソースコードをみたら、現在の実装は、正規表現をつかいまくって HTML を解析しているため、何かイレギュラーなことがあった場合には、動作しなくなります。該当する関数は emacs-goodies-el:/elisp/debian-el/debian-bug.el(debian-bug-build-bug-menu) です。

BTS のフォーマットが変わったので、何かうごかなくなっていないかと心配していましたが、特にうごかないということはないようです。

みると submitter のメールアドレスがうまく解析できなかった場合には 'thanks to XXXX (closes: XXXX)' が追加されないという仕様になっています。たまにこれが発生するので、再現する条件をさがしてバグを直したいですね。

---

\*23 <http://bugs.debian.org/207852>

```

(with-temp-buffer
 (message "Fetching bug list...")
 (call-process "wget" nil '(t t) nil "--quiet" "-0" "-"
  (concat
   "http://bugs.debian.org/cgi-bin/pkgreport.cgi?src="
   package))
 (message "Fetching bug list...done")
 (goto-char (point-min))
 (while
  (re-search-forward
   "\\(<H2.*</a>\\(.+\\)</H2>\\)\\|\\(<li><a
href=\\\"\\(bugreport.cgi\\?bug=\\([0-9]+\\)\\)\\\">\\(#[0-9]+: \\(.+\\)\\)\\</a>\\)"
   nil t)
  (let ((type (match-string 2))
        ;;(URL (match-string 4))
        (bugnumber (match-string 5))
        (description (match-string 6))
        (shortdescription (match-string 7)))
    (cond
     (type
      (setq bugs-are-open-flag (not (string-match "resolved" type)))
      (save-excursion
       (set-buffer debian-bug-tmp-buffer)
       (insert "\\-\\n\\n" type "\\n")))
     (t
      (setq bug-alist (cons (list bugnumber description) bug-alist))
      (when bugs-are-open-flag
       (when (and (re-search-forward
                  "Reported by: <a class=\\\"submitter\\\"
href=\\\"pkgreport.cgi\\?submitter=[^;]+;arch=source\\\">
                  nil t)
                 (or (looking-at "&quot;\\(.*\\)&quot; &lt;")
                     (looking-at "\\(.*\\) &lt;")))
                (setq shortdescription
                  (concat "Bug fix: \\\"" shortdescription
                          "\\", thanks to "
                          (debian-bug-rfc2047-decode-string
                           (match-string 1))
                          " (Closes: #\" bugnumber ").")))
                (setq bug-open-alist
                  (cons
                   (list bugnumber shortdescription)
                   bug-open-alist))))
      bug-open-alist)))

```

実際に changelog に追加する部分は emacs-goodies-el:elisp/dpkg-dev-el/debian-changelog-mode.el(debian-changelog-close-bug) にあります。

## 7 Debian とステータス

えとー



## 7.1 statoverride とは？

ステータス情報を操作するインターフェースファイル、ディレクトリ、デバイスファイルなど dpkg が扱えるファイルシステムオブジェクトならなんでも扱うことができる。

## 7.2 歴史

dpkg-statoverride の前には dh.suidregister というのがあった。その suidregister の幾つかの問題を解決したものとなっている。2000 年のことなのですでに 5 年以上の歴史のある古いものになっているが今だ有効な仕組み。<sup>\*24</sup>

## 7.3 どんな風に使われている？

## 7.3.1 メンテナ編

まずは身近な postfix で用いられているメンテナスクリプトから抜粋してみましょう。

postinst

```
1 dpkg-statoverride --remove \${POSTDROP} >/dev/null 2>&1 || true
2 dpkg-statoverride --remove /var/spool/postfix/public >/dev/null 2>&1 || true
3 dpkg-statoverride --remove /usr/sbin/postqueue >/dev/null 2>&1 || true
4 dpkg-statoverride --update --add root postdrop 02555 \${POSTDROP}
5 dpkg-statoverride --update --add postfix postdrop 02710 /var/spool/postfix/public
6 dpkg-statoverride --update --add root postdrop 02555 /usr/sbin/postqueue
```

postrm

```
1 dpkg-statoverride --remove /usr/sbin/postdrop >/dev/null 2>&1 || true
2 dpkg-statoverride --remove /var/spool/postfix/public >/dev/null 2>&1 || true
3 dpkg-statoverride --remove /usr/sbin/postqueue >/dev/null 2>&1 || true
```

1. dpkg-statoverride --remove /usr/sbin/postdrop *i* /dev/null 2*i*&1 — true  
/usr/sbin/postdrop の statoverride の設定を削除する
2. dpkg-statoverride --remove /var/spool/postfix/public *i* /dev/null 2*i*&1 — true  
/var/spool/postfix/public の statoverride の設定を削除する
3. dpkg-statoverride --remove /usr/sbin/postqueue *i* /dev/null 2*i*&1 — true  
/usr/sbin/postqueue の statoverride の設定を削除する
4. dpkg-statoverride --update --add root postdrop 02555 \$POSTDROP  
ユーザ「root」グループ「postdrop」で実行時にグループ ID のユーザで起動、読み取りと実行をオーナーユーザ、オーナーグループ、その他に許可したものを /usr/sbin/postdrop に付与する。
5. dpkg-statoverride --update --add postfix postdrop 02710 /var/spool/postfix/public  
ユーザ postfix グループ postdrop で実行時にグループ ID を指定した、オーナーユーザ

\*24 参考リンク:<http://lists.debian.org/debian-dpkg/2000/06/msg00015.html>

には読み、書き、実行を、オーナグループには実行を、その他にはなにも出来ない権限を/var/spool/postfix/public に付与する。

6. dpkg-statoverride --update --add root postdrop 02555 /usr/sbin/postqueue  
ユーザ root グループ postdrop で実行時にグループ ID を指定した、読み取りと実行をオーナユーザ、オーナグループ、その他に許可したものを/usr/sbin/postqueue に付与する。

以上のように、プログラムにデフォルトのパーミッションを付与したい場合に用いることができる。

パッケージメンテナがこれを利用したほうがよいのには理由がある。

debian のポリシーマニュアルでは標準のパーミッションなどが規定されている、しかし、規定されているものでは却って不便になってしまうものがあり、ポリシーから離れてパーミッションなどを変更する場合にあとからユーザがポリシーにない状態の確認を容易にするために用いるインターフェースとして利便性のためである。

別の例を上げてみよう。

```
for i in /usr/bin/foo /usr/sbin/bar
do
  if ! dpkg-statoverride --list $i >/dev/null
  then
    dpkg-statoverride --update --add sysuser root 4755 $i
  fi
done
```

解説 - /usr/bin/foo や /usr/sbin/bar に statoverride の設定がされていなければユーザ「sysuser」、グループ「root」、パーミッション「4755」を設定する。

第二に postinst などでは chmod, chown などを使う場合でもユーザの都合でオーナーやパーミッションを変更させることがありうるが、たとえ変更したとしてもパッケージの再インストール時やアップグレード時などにパッケージ標準のオーナーとパーミッションに上書きされてしまう可能性がある。これを防ぐためにパッケージ標準のオーナー及びパーミッションを設定可能にしておくのがよいと思われる。

パーミッションをごたごた言う BTS を減らすためにも statoverride は活用しましょう！

### 7.3.2 ユーザ編

サーバ管理者などはパッケージにより提供されているオーナーやパーミッションでは目的が達成できない場合がある、アップデートの少ない stable を使ったとしてもセキュリティアップデートなどで update がかった場合にどうしても上書きされてしまい、初期の値に戻されてしまうことになる。これを回避するためには、dpkg-statoverride を使うことによりパッケージの上書きを回避してユーザ設定のオーナーとパーミッションをアップグレードしても無関係に用いることができるようになる。

GUI 環境がある場合は dsys を使ってみてください。

## 7.4 使い方

### 7.5 一般的な用法

#### 1. ステータスの追加

```
# dpkg-statoverride --add ユーザ名 グループ名 パーミッション ファイル名
```

#### 2. ステータスの即時追加]

```
# dpkg-statoverride --update --add ユーザ名 グループ名 パーミッション ファイル名
```

#### 3. ステータスの削除

```
# dpkg-statoverride --remove ファイル名
```

#### 4. ステータスの変更

```
# dpkg-statoverride --remove ファイル名 # dpkg-statoverride --update --add ユーザ名 グループ名 パーミッション ファイル名
```

#### 5. ステータスの確認

```
# dpkg-statoverride --list パターン
```

### 7.6 マニアックな用法

`--admindir` オプションを使いホームディレクトリの `.` (ドット) ファイルのパーミッションを制御する。

### 7.7 パッケージアップグレード

#### 1. `postinst` などで `statoverride` が使われているものを管理者などが `statoverride` の設定を変更した場合

`postinst` などで設定を `--remove` してから `--add` している場合などにはシステム管理者が `statoverride` を設定しても `postinst` などの設定に書き換えられてしまう。(postfix など)

`postinst` などで `--add` のみの場合には管理者などが設定した `statoverride` の設定になる

#### 2. `postinst` などで `statoverride` をセットされているが管理者が `chown`、`chmod` などをした場合

`statoverride` の設定に従ってパーミッションが変更される

#### 3. `postinst` などで `chown` や `chmod` しているものを管理者などが `statoverride` の設定を変更した場合

システム管理者が `statoverride` を設定してもパッケージの `postinst` などの `chown`、`chmod` で書き換えられてしまう。

#### 4. `postinst` などでは `statoverride` も `chown`、`chmod` も使われていないものを `statoverride` の設定を行なった場合

`statoverride` の設定に従ってファイルのパーミッションが変更される

#### 5. `postinst` などで `chown` や `chmod` しているものを管理者などが `chown` や `chmod` で設定を変更した場合

システム管理者が `chown` や `chmod` で設定してもパッケージの `postinst` などの `chown`、`chmod` で書き換えられてしまう。

## 7.8 おわりに

statoverride は alternatives と比べると単純な機能ではシンプルなものだし知名度は更に低い  
が、メンテナもユーザも楽になれるツールなので是非活用していただきたいと思う。

## 8 Debian Weekly News 翻訳フロー

小林儀匡



### 8.1 はじめに

Debian Weekly News (DWN) という、Debian コミュニティのために毎週発行されるニュースレターを御存知でしょうか。Debian 通になるためには読むのが必須とされている(?) ニュースです<sup>\*25</sup>。Debian JP の debian-users メーリングリスト<sup>\*26</sup>を購読しているかたなら、最近では毎週のように翻訳したものが流れてくるので知っているでしょう。この DWN 翻訳作業は、以前は今井伸広さんが一人でされていましたが 2005 年 6 月ごろから DDTTP<sup>\*27</sup>日本語チームコーディネータの田村一平さん、そして小林がチームを組んでおこなっています。ここでは、その翻訳作業の流れについて説明します。

### 8.2 Debian Weekly News とは

DWN は Debian コミュニティのための週刊ニュースレターで、UTC で毎週火曜 18:45 ごろ (JST で毎週水曜 3:45 ごろ) に発行されます。「週刊」というだけあって基本的に毎週発行されますが、発行されないことも年に数回あります。発行形態はメーリングリストとウェブページの 2 種類あり、前者は本家の debian-news メーリングリスト<sup>\*28</sup>で購読できます (もちろんこのメーリングリストには DWN の他に不定期のニュースも珠に流れてきます)。また後者は <http://www.debian.org/News/weekly/> で参照できます。様々なメーリングリストでの話題やイベントについて書かれているので、読んでおくと Debian 界隈のニュースに (無駄に) 詳しくなれます。

DWN の編集は最近ではほとんど Martin 'Joey' Schulze 一人の作業に委ねられています。しかしフッタに書かれる編集者情報を見る限り、2 号に 1 号くらいは他の人が助けているようです。

ちなみに創刊号は 1999 年 1 月 4 日に出され、その編集者は Joey Hess でした。記念すべき第 1 号の editorial によれば、Linux Weekly News<sup>\*29</sup>を真似て作られたようです。また editorial の次に書かれた記事第 1 号のタイトルは 'RMS is using Debian.' です。

---

<sup>\*25</sup> もちろん、きちんとすべて頭に入れ、忘れないよう毎日読み返す、なんてことはしなくてかまわないでしょう。ただ、世界に広がる Debian コミュニティにはどんな人がいて、どんな活動をしているか、Debian がどのような人々の日常的な保守作業や議論を通して作られているかは、一ユーザーとして知っておいたほうがよいように思うので、目は軽く通しておいたほうがよいと思います。あくまで私見ですが。

<sup>\*26</sup> [debian-users@debian.or.jp](mailto:debian-users@debian.or.jp)。

<sup>\*27</sup> Debian Description Translation Project. Debian パッケージ説明文の翻訳をするプロジェクト。

<sup>\*28</sup> [debian-news@lists.debian.org](mailto:debian-news@lists.debian.org)。

<sup>\*29</sup> <http://lwn.net/>。

## 8.3 ファイル形式 – WML

ウェブページ版の DWN は Debian のサイトの一部なので、他のページと同じ CVS リポジトリ<sup>\*30</sup>、そして同じ WML というファイル形式を利用しています。

ファイル形式には他のページと同様に WML が用いられています。WML とはウェブサイトメタ言語 (web site meta language) のことで、Debian では wml パッケージとして提供されています。簡単に言ってしまうと HTML に命令を混ぜたような言語で、例えばウェブページのコンテンツに定型的なヘッダ・フッタ、さらに HTML の head を加えたりするのに用いられています。また、条件分岐を用いた、「 のリリース前にはこのコンテンツを表示し、リリース後にはこのコンテンツを表示する」というような表示の切り替えや、Debian のウェブページの翻訳版がオリジナル版のどのバージョンに基いているかを管理するのにも用いられています。WML についてより詳しく知りたければ [http://www.debian.org/devel/website/using\\_wml](http://www.debian.org/devel/website/using_wml) を参照してください<sup>\*31</sup>。

### 8.3.1 DWN の WML

Debian Weekly News の場合だと WML の形式は決まっています。以下では (わざわざ説明する必要もない、見れば分かるものだと思いますが) DWN 2005 年第 44 号のものを例にとって説明します。

まず最初に命令が決まります。ここには、発行日時・そしてアーカイブのページに表示されるサマリ (主だった記事のキーワードのリスト) が書かれます。その後に CVS の Id が書かれています。

```
#use wml::debian::weeklynews::header PUBLISHDATE="2005-11-01" SUMMARY="Dependencies, OpenSSH"
# $Id$
```

その後で、‘Welcome to this year’s nth issue of DWN, the weekly newsletter for the Debian community.’ で始まる editorial が来ます。

```
<p>Welcome to this year’s 44th issue of DWN, the weekly newsletter for the
Debian community. Nathanael Nerode <a
href="http://lists.debian.org/debian-devel/2005/10/msg00388.html">reported</a>
that current GCC versions support the old i386 processor again and hence
Debian could retain i386 compatibility in the upcoming <a
href="$(HOME)/releases/etch/">etch release</a>.</p>
```

あとは普通の記事の繰り返しです。

```
<p><strong>Calculating Development Package Dependencies.</strong> Jay
Berkenbilt <a
href="http://lists.debian.org/debian-devel/2005/10/msg00184.html">\
proposed</a> to work on a <a
```

---

<sup>\*30</sup> リポジトリ URL は `:pserver:anonymous@cvs.debian.org:/cvs/webwml` で、ウェブインタフェースも `http://cvs.debian.org/?root=webwml` で利用できます。この中の webwml ディレクトリに `http://www.debian.org/` 用の wml ファイルやそれからウェブページをビルドするツールなどが含まれています。例えば DWN 2005 年第 44 号英語版は `webwml/english/News/weekly/2005/44/index.wml` にあり、その日本語版は `webwml/japanese/News/weekly/2005/44/index.wml` にあります。

<sup>\*31</sup> ちなみに筆者は、WML が Debian 以外にどこで使われているか知りません

[debhelper](http://packages.debian.org/debhelper) script that helps calculating [libtool](http://packages.debian.org/libtool) dependencies for development packages. Goswin von Brederlow [pointed out](http://lists.debian.org/debian-devel/2005/10/msg00519.html) that with [\multiarch](http://raw.no/debian/amd64-multiarch-2) there may be concurrent `.la` files to handle. No consensus in favour of such a script was reached.

Junichi Uekawa (19978;24029;32020;19968;)

<a

[\mentioned](http://lists.debian.org/debian-devel/2005/10/msg00316.html) the [d-shlibs](http://packages.debian.org/d-shlibs) package that contains scripts to support the maintainer in this regard.</p>

以上が DWN の記事部分です。

記事部分のあとには、箇条書で書けるような様々な情報のコーナーが続きます。主に前号の発行以降に変化があったパッケージについての情報で、最近載っているのは

‘Security Updates.’ (「セキュリティ上の更新。」) DSA 番号・パッケージ名・脆弱性についての記述) のリスト。

‘New or Noteworthy Packages.’ (「新規または注目すべきパッケージ。」) パッケージ名・パッケージ説明文のリスト。

‘Orphaned Packages.’ (「みなしご化されたパッケージ。」) パッケージ名・パッケージ説明文・debbugs のバグ番号のリスト。

‘Removed Packages.’ (「削除されたパッケージ。」) パッケージ名・パッケージ説明文・パッケージ削除理由のリスト。

の 4 コーナーですが、このうち「削除されたパッケージ。」は比較的新しい項目です。去年は、Debian Package a Day’s Journal<sup>\*32</sup>で紹介されたパッケージのリストを含む ‘Debian Packages introduced last Week. Ever’ (「先週紹介された Debian パッケージ。」) というコーナーがあったのですが、これは、ウェブページが更新されなくなった (さすがにネタが尽きた?) ために廃止されたようです。以下にコーナーの例を示します。

<p><strong>Security Updates.</strong> You know the drill. Please make sure that you update your systems if you have any of these packages installed.</p>

<ul>

<li>DSA 872: [koffice](#) --  
Arbitrary code execution.

<li>DSA 873: [net-snmp](#) --  
Denial of service.

<li>DSA 874: [lynx](#) --  
Arbitrary code execution.

<li>DSA 875: [openssl094](#) --

---

\*32 <http://www.livejournal.com/users/debaday/>.

Cryptographic weakness.  
<li>DSA 876: <a href="\$(HOME)/security/2005/dsa-876">lynx-ssl</a> --  
Arbitrary code execution.  
<li>DSA 877: <a href="\$(HOME)/security/2005/dsa-877">gnump3d</a> --  
Several vulnerabilities.  
<li>DSA 878: <a href="\$(HOME)/security/2005/dsa-878">netpbm-free</a> --  
Arbitrary code execution.  
</ul>

最後に定型メッセージが入ります。

<p><strong>Want to continue reading DWN?</strong> Please help us create this newsletter. We still need more volunteer writers who watch the Debian community and report about what is going on. Please see the <a href="\$(HOME)/News/weekly/contributing">contributing page</a> to find out how to help. We're looking forward to receiving your mail at <a href="mailto:dwn@debian.org">dwn@debian.org</a>.</p>

そしてフッタ用の WML 命令が入ります。編集者の名前を書くのに使われます。

```
#use wml::debian::weeklynews::footer editor="Martin 'Joey' Schulze"
```

## 8.4 翻訳作業

翻訳作業はだいたい以下のような流れになっています。

週末 Martin 'Joey' Schulze がプライベートな草稿<sup>\*33</sup>に記事を追加する。

水曜未明 オリジナル版がリリースされる。

水曜 今井さんが翻訳チームの田村さん・小林に作業を割り振る。

水曜～日曜 翻訳チームメンバーの3人が個々に翻訳し、Debian JP の debian-www メーリングリスト<sup>\*34</sup>にそれを投稿して査読をお願いする。それに対して杉山さん・武井さんなどが査読をしてくださる。

月曜～火曜 (基本的に今井さんが) debian-users に日本語版をリリースする。

### 8.4.1 Martin 'Joey' Schulze のオリジナル版リリース作業

Martin 'Joey' Schulze は、オリジナル版をリリースするために独自のリポジトリ<sup>\*35</sup>を用いています。彼はそれに記事を追加していき、UTC で月曜日の夕方に、「ごく親しい関係者」向けに DWN のプレビューをリリースします。彼はこのプレビューを dwn@debian.org に送り、DWN チームのレビューを受けます。またプレビューは、早目に翻訳にとりかかりたい人のために dwn-trans@debian.org にも送られます。こうして、いくつかの言語ではオリジナル版の正式なリリースと同時に翻訳版をリリースすることができるようになっていきます。日本語版は現在、正式リリースを待って作業を開始しています。

<sup>\*33</sup> <http://www.infodrom.org/~joey/Writing/DWN/>.

<sup>\*34</sup> [debian-www@debian.or.jp](mailto:debian-www@debian.or.jp).

<sup>\*35</sup> `:pserver:anonymous@cvs.infodrom.org:/var/cvs/infodrom.org/public_html/src/Writing/DWN`.

#### 8.4.2 割り振り

翻訳チームメンバー 3 人への作業の割り振りは基本的に今井さんが行います。チーム制を始めたばかりのころは「記事部分前半」・「記事部分後半」・「それ以外 (様々なコーナー)」という分け方をしていました。しかし、記事というのは流れがあるので訳していてそれなりに楽しいのですが、「それ以外 (様々なコーナー)」は、脆弱性・パッケージ削除理由といった定型文句が多数を占めるものがある一方で、流れがなくて訳しづらいパッケージ説明文もあります。(どちらにしるローテーション制なのですが) それだとやりがいに差が出るためか<sup>\*36</sup>、最近では「記事部分の 1/3 とどれかのコーナー」という分け方になっているようです。

#### 8.4.3 翻訳・査読依頼

翻訳作業は、CVS リポジトリからオリジナル版の DWN の wml ファイルをコピーして始めます。このときファイルのコピーには、同じリポジトリに入っている webwml/copypage.pl を使います。というのも、この Perl スクリプトはコピー元のオリジナル版 wml ファイルに含まれる CVS の Id キーワードの値から翻訳バージョンチェック用の WML 命令を作り出してくれるからです。またこのスクリプトは、オリジナル版 wml ファイルに含まれている Latin-1 文字コードの文字を適切な文字実体参照に置き換え、ASCII だけを含むファイルにしてくれます。ウェブインタフェースを使ったダウンロードや、cp コマンドによる生のコピーでは、それらがおこなわれません。

実際の翻訳作業では、査読のしやすさを考えて、オリジナルの記事・リスト項目を残し、その後ろに翻訳をつけていきます。「この文の意味がよく分からない」「この単語はこう解釈した」といったメモを査読者に残したい場合は、「#」でコメントアウトしたものをオリジナルと翻訳の間に挟んで残します。

#### 8.4.4 日本語版リリース作業

リリース作業は次のような手順でおこないます。

1. Debian JP debian-www メーリングリストに送られた 3 人の翻訳を wml ファイルにマージする。
2. 査読をしてくださった方々のコメントを反映させる修正をおこなう。
3. w3m で表示させ、査読でも指摘されずに残った明らかなミスがないか、自分の目で簡単にチェックする。
4. w3m の表示を見ながら、全角文字同士や全角文字と ASCII の間に不必要な空白ができていないか、または ASCII の周りに空白がきちんとあるかを調べ、必要に応じて改行位置やタグの位置を変える。詳しくは 8.5 を参照のこと。
5. 最下行にあるフッタ用 WML 命令に、editor フィールドとは別に translator フィールドをつけることができるので、そこに翻訳者の名前を連ねる。
6. 査読しやすいように残してあったオリジナルの記事・リスト項目と翻訳時のメモを消す (これでウェブ用翻訳版 wml ファイルは完成となる)。
7. 今井さんの Ruby スクリプトで、wml ファイルを Debian JP debian-users メーリングリスト投稿用プレインテキストに変換する。
8. &ouml; などの文字実体参照を w3m などの出力に合わせて変換する。

<sup>\*36</sup> 本当の理由は知らないのですが、もしかしたら違うかもしれません。

9. Emacs の auto-fill-mode を利用して適当なところで改行する。ただし、リンクの代わりとなる URL リスト内の項目番号を示す「[1]」のような文字列の前で改行されてしまった場合は、それを手動で前の行末に戻す。
10. 以前の debian-users への投稿を元に体裁を整える。
11. コメントを加える。大変な査読作業をしてくださった方々への謝辞を忘れないようにする。
12. 完成した投稿用プレインテキストを debian-users に送信する。
13. 最後にウェブ用翻訳版 wml ファイルを debian-www に送信し、コミットをお願いする（もちろん、コミット権のある今井さんが作業者の場合は直接コミットとなる）。

## 8.5 翻訳作業時の注意点

翻訳作業時には以下のような点に注意を払います。

- ウェブページの見栄えを意識して、ASCII と全角文字の間には空白を入れる。
- 変なところで改行すると、テキストブラウザで見たときに全角文字どうしの間に空白が入ってしまうことがある。それを防ぐため改行位置には注意を払い、たとえば次のような方法をとる。
  - 見かけに反映されることのない、タグの内側での改行を多用する。
  - 特にタグ直後の改行には注意する。
  - 結果的にスペースを伴うような改行がどうしても必要なときは、行末に\を入れる。
- 改行位置に自信がないときは w3m で見てみる<sup>\*37</sup>。

## 8.6 今後の課題

今後の課題としては以下のようなものがあるでしょう。中には DWN の翻訳に留まらず他の翻訳にも関係するものもあります。

**定型文句の翻訳の自動化** 定型文句を翻訳するときには、訳語や表現形式を合わせるために、その文句が使われている過去の翻訳を探して copy and paste するという作業が必要になります。この作業は、オリジナル版と翻訳版を行き来して該当部分を探し出さなければならないので、案外面倒なものです。対訳をデータベース化し、そのデータを用いて翻訳できる部分は作業前に予め機械的に翻訳できるようにしておくと、もう少し効率よく作業を進められる気がします。

**対訳表の保守** Debian JP に対訳表<sup>\*38</sup> があるのですが、現在は更新されていません。APT 関連、Debian プロジェクト関連、あるいはその他の Debian でよく使われる用語の対訳表<sup>\*39</sup> を整備できたらと思います。新規に翻訳作業に携わりたい人にとって、対訳が簡単に調べられることはかなり重要でしょう。

---

<sup>\*37</sup> -T text/html とオプションを指定すれば、wml ファイルを HTML ファイルとして見ることができます。

<sup>\*38</sup> HTML 版が [http://www.debian.or.jp/Documents/trans\\_table/trans\\_table.html](http://www.debian.or.jp/Documents/trans_table/trans_table.html) にあり、dict 形式のものも <http://www.debian.or.jp/devel/doc/about-trans-table.html> で手に入ります。

<sup>\*39</sup> 翻訳者を煩わせるサードパーティのアプリケーション名の表記に関するポリシーも一緒に含まれていたらよいと思います。

## 8.7 おまけ: 翻訳作業に有用なツール

小林が翻訳作業によく用いているのは、英辞郎<sup>\*40</sup>、NDTPD<sup>\*41</sup>、Lookup<sup>\*42</sup>の組み合わせです。英辞郎のデータを FreePWING<sup>\*43</sup>で変換して得られたデータに、Emacs 上のクライアント Lookup からサーバ NDTPD を経由してアクセスします。

また、文書の変更を管理するのに、バージョン管理システムとして Subversion<sup>\*44</sup>を利用しています。差分を unified diff で表示しても分かりにくいときは、DocDiff<sup>\*45</sup>で表示させると分かりやすくなることがあります。

---

<sup>\*40</sup> <http://www.eijiro.jp/>。

<sup>\*41</sup> <http://www.sra.co.jp/people/m-kasahr/ndtpd/>。Debian パッケージは <http://packages.debian.org/ndtpd>。

<sup>\*42</sup> <http://openlab.ring.gr.jp/lookup/>。Debian パッケージは <http://packages.debian.org/lookup-el>。

<sup>\*43</sup> <http://www.sra.co.jp/people/m-kasahr/freepwing/>。Debian パッケージは <http://packages.debian.org/freepwing>。

<sup>\*44</sup> <http://subversion.tigris.org/>。Debian パッケージは <http://packages.debian.org/subversion>。

<sup>\*45</sup> <http://www.kt.rim.or.jp/~hisashim/docdiff/>。Debian パッケージは <http://packages.debian.org/docdiff>。

## 9 Debian Weekly News trivia quiz

上川 純一



ところで、Debian Weekly News (DWN) は読んでいますか？Debian 界隈でおきていることについて書いている Debian Weekly News. 毎回読んでいるといろいろと分かって来ますが、一人で読んでいても、解説が少ないので、意味がわからないところもあるかも知れません。みんなで DWN を読んでみましょう。

漫然と読むだけではおもしろくないので、DWN の記事から出題した以下の質問にこたえてみてください。後で内容は解説します。

### 9.1 2005 年 27 号

<http://www.debian.org/News/weekly/2005/27/> にある 2005 年 7 月 5 日版です。

問題 1. 共有ライブラリを作成するために必要な gcc のコンパイルオプションは

- A -fPIC だが i386 などでは-fPIC でなくても動作するという例外がある
- B -shared-library というのがある
- C -ansi

問題 2. sarge をリリース後の sid にて一番壊れている debian-installer の部品は

- A deboootstrap
- B joeyh
- C cdrom

問題 3. Lars Wirzenius は RFP が多すぎる、と懸念していた。現在有効な RFP はいくつあるか

- A 1000 以上
- B 100 くらい
- C 500 くらい

問題 4. Firefox と Thunderbird についての商標ポリシーは何が問題か

- A 名前は特許です
- B 変更を加えるとその名前を名乗る事ができない
- C Debian が嫌い

問題 5. glade1 を glade2 に移行することで発生する問題はなにか

- A アプリが重くなる
- B glade2 への移行は gnome2 への移行が必要になるが、まだすべてのアプリケーションが gnome2 に移行できる準備ができていないわけではない
- C カーネルのバージョンが 2.6 以上必須になる

問題 6. 教育上不適切だと思われる画像を表示するスクリーンセーバの利用についての質問への回答として Larz が勧めたのは

- A 眼鏡の度数を下げろ
- B モニターのかわりにプロジェクターを使え
- C スクリーンセーバは画面を暗転するものだけをデフォルトとしてはいれよう

問題 7. DAK でバージョン文字列の中に `が入ったパッケージを扱う際に問題となるのは

- A 内部で別の目的でセパレータ文字列として利用しているため、それと衝突する
- B チルダは主義主張に反するので扱えない
- C 問題はない

## 9.2 2005 年 28 号

<http://www.debian.org/News/weekly/2005/28/> にある 2005 年 7 月 12 日版です。

問題 8. Matt Taggart が提示した LSB 3.0 に準拠するのに必要な点はなにか

- A 新しい glibc と xorg パッケージが必要になる
- B 根性
- C 気合い

問題 9. gcc 4.0 を新規にデフォルトコンパイラとして導入することで、何が推奨されたか

- A C++ 関連のパッケージについてはアップロードを控える
- B C++ で書いてあるプログラムは C で書き直す
- C C++ で書いてあるプログラムをできるだけ java のプログラムでおきかえる

問題 10. Ludovic Brentaさんは ada について

- A 一年半関連したパッケージをごっそりとメンテナンスし、管理用ポリシーを書き上げた
- B 上流の開発をしていた
- C Debian Developer としてメンテナンスした

問題 11. パッケージの循環依存についてなにが言えるか

- A 循環依存はできるだけ使うべき
- B 循環依存はおいしい
- C 循環依存はアップグレード処理に際してうまくあつかえないため、直すべき。Pre-Depends を使えば問題を一部解消できる

問題 12. Frank Lichtenheld さんは何を計画したか

- A Debian から non-free なドキュメントを削除する計画
- B Debian から一定の人種を排除する計画
- C Debian から一部のライセンスを排除する計画

### 9.3 2005 年 29 号

<http://www.debian.org/News/weekly/2005/29/> にある 2005 年 7 月 19 日版です。

問題 13. debian-cd でこれから改善するという 11 の項目にはいついたのは

- A 壊れた依存関係をうみださないようにチェックを追加したい
- B CD の素材はリサイクル可能なものに限定する
- C CD の大きさの規格を変更する

問題 14. GNU Hurd の現状について

- A buildd が稼働していて、40% 程度のパッケージがビルドできている。
- B 誰も使っていないので動くのかよくわからない
- C 今後カーネルは Linux カーネルを使うことになった

問題 15. g++4.0 への移行にともなって c++ で作成した共有ライブラリには何が必要となるか

- A ABI が変更となるので、SONAME がかわっていてもパッケージ名に 'c2' を追加して変更する
- B 特になにもないのでそのままにしておく
- C ユーザへのいやがらせのために意味もない依存を負荷すること

問題 16. バグ追跡システムへ追加された機能は何か

- A 過去のバグレポートの傾向をみて重みをつけてくれるシステムが追加された
- B バグ報告の内容から類似のバグを検索して解決策候補を提示してくれるシステムが追加された
- C どのバージョンでバグが存在しているのかを調べるバージョントラッキングが追加された

問題 17. 印刷関連の問題を解決するべく発足したのは

- A debian-printing メーリングリスト
- B プリンタページ
- C 印刷会議

### 9.4 2005 年 30 号

<http://www.debian.org/News/weekly/2005/30/> にある 7 月 26 日版です。

問題 18. piuparts は何をするツールか

- A Debian パッケージのアップグレードなどを実地検証するツール
- B  $\pi$  を計算するツール
- C ゲーム

問題 19. cpufrequtils を利用してデフォルトの CPU governer をシステム起動のはやい時期に変更することに対する反論は

- A 起動シーケンスの初期には CPU はどうせ十分働く必要があるし、最初の起動の一分に必要な CPU による電力消費はたいした量ではない
- B CPU の周波数なんて関係ないんです
- C ノートパソコンなぞ使うな

問題 20. エディタ (jed) のデフォルトの外観を設定する debconf の質問を追加することに関しての反応はどうだったか

- A エディターがデフォルトでどういう外観であるのかということはシステム管理において重要な設定項目なのでシステム管理者に質問すべき
- B jed にデフォルトの外観なんて無い
- C そんな質問をシステムアドミニストレータにする必要はないだろう。

問題 21. デーモン管理用のパスワードをどうやって管理者に通知すべきかという質問に対しての回答は

- A デーモン管理にパスワードは使うな
- B ユーザに設定させる
- C /etc/以下に適切な権限で配置したファイルに記述しておく

問題 22. Debian ロゴに使われている文字のフォントは

- A Poppl Laudatio Condensed という商用フォント
- B monafont
- C kochi

問題 23. BTS へのバグのサブスクリプションが可能になった。どうやったらバグレポートにサブスクライブできるか

- A owner@bugs にメールを送る
- B IRC で kamion をつつく
- C XXXX-subscribe@bugs.debian.org にメールを投げる

問題 24. texi2html の新しい挙動はなにか

- A セクション毎にファイルを生成するようにすると、サブディレクトリにファイルを生成するようになる。
- B 毎回エラーで終了する
- C html ではなく xhtml を生成する

問題 25. debtags はどうなったか

- A Debian パッケージ全部にタグをつけるのが完了した
- B Packages ファイルに統合された
- C この世の中から排除された

## 9.5 2005 年 31 号

<http://www.debian.org/News/weekly/2005/31/> にある 2005 年 8 月 2 日版です。

問題 26. パッケージの description についてレビューをしようというプロジェクトがはじまった。何に関して議論が収束していないか

- A description は本当に必要なのか
- B どれくらいの技術的な内容を description に含めるべきか。
- C どのパッケージを対象にするか

問題 27. Popularity Contest のレポートの送信で現在圧縮して送信してくれるのはどのプロトコルを利用した場合か

- A SSTP
- B HTTP
- C メール

問題 28. hmh が主張している next generation initscripts はいつごろから始まったプロジェクトか

- A Debconf5
- B Debconf0
- C Debconf2

問題 29. メーリングリストアーカイブのウェブページに最近追加された機能は何か

- A 類似検索機能
- B SPAM 報告機能
- C メールを読んだ気にさせる機能

## 9.6 2005 年 32 号

<http://www.debian.org/News/weekly/2005/32/> にある 2005 年 8 月 9 日版です。

問題 30. あるパッケージをアップロードした場合の debian に与えるリスクを計算する方法が欲しいというリクエストに対して出て来た、現在どのパッケージがいちばん testing に影響しているというのを調査するには

- A グーグルで検索する
- B <http://bjorn.haxx.se/debian/stalls.html> にある どのパッケージがどれくらいのパッケージの testing 入りを邪魔しているかというページを見る
- C `ls -l /`

問題 31. GNUStep で問題なのは何か

- A FHS に全く準拠していなく、GNUStep をそのまま FHS に準拠させるのは難しい。
- B 使いにくい
- C 動かない

問題 32. Ian Jackson は、「Debian」 Core Consortium について何を宣言したか

- A 我々が Debian をのっとした
- B Debian は使えない
- C Debian Core Consortium という名前は正式名称ではないので問題無い

問題 33. MySQL 4.0 から 4.1 への移行についてどういうことが議論されたか

- A libmysqlclient の移行のためにバグをファイルするのは、今は gcc 4.0 の移行中なため、避けて欲しい。
- B MySQL はもうすてよう
- C PySQL に名前が変わりました

問題 34. Andrews Barth は gnome2.10 を etch にいれよう、testing に入れるためにアップロードをしばらくひかえよう、とかげごえをかけましたが、出た反論が

- A gnome2.10 なんて誰もつかっていない
- B Nathanael Nerode さんが言うには、xorg の移行によってブロックされてしまっているの  
しばらくは無理
- C testing なんて存在しません

問題 35. Helen Faulkner さんが作成したメーリングリストは

- A debian-science メーリングリスト
- B debian-helen メーリングリスト
- C debian-faulkner メーリングリスト

問題 36. xorg6.9 について David Nusinow が報告したのは

- A quilt ベースのパッチシステムに移行したおかげで以前までは数週間かかっていたパッチの移行が 3-4 日しかかからなくなりました。
- B 画面表示のいろあいが変わりました
- C 高速化しました

## 9.7 2005 年 33 号

<http://www.debian.org/News/weekly/2005/33/> にある 2005 年 8 月 16 日版です。

問題 37. Debian の 12 回目の誕生日はいつか

- A 2005 年 8 月 16 日
- B 2006 年 8 月 16 日
- C 2005 年 4 月 1 日

問題 38. RC バグの影響で testing からパッケージを削除される場合がある。その場合にどうしたら testing にパッケージが戻るか

- A リリースマネージャに賄賂をおくる
- B リリースマネージャを人質にとる
- C RC バグを全部修正する。

問題 39. カーネルのソースパッケージは linux-source-2.6.12 になっており、ソースの入ったバイナリパッケージは linux-2.6 になった。その理由は

- A ピリオドの消費を節約するため
- B ユーザを混乱させるため
- C 古いバージョンをアーカイブ内に維持しつつユーザのアップグレードを簡単にするため

問題 40. Debian メンテナはバグを上流に報告する義務がある、という点に関して、Eric Dorland が反論した理由は

A firefox パッケージには 300 くらいのバグが現在 open されており、上流にフォワードするという作業だけで時間が無駄にとられてしまう。

- B BTS なんか見てない
- C 自分では使っていないので使っている人が報告してほしい

問題 41. Joerg Jaspert が発表した、Debian Developer の追放の規則によると

- A 追放期間は一年間で、保釈金を支払うともどつてこれる
- B DPL のわるぐちを言った Debian Developer は追放できる
- C 一旦追放されても NM プロセスを経て Debian Developer にもどつてこれる

問題 42. LinuxFund が Debian に対して実施すると発表したのは

- A 月 \$ 5 0 0 の資金提供を 1 年間実施する。
- B Debian DVD を売る
- C Debian を使う

問題 43. Hanna Wallach が Debian Women の現状について懸念を示した内容としては

A Debian Project をのつとめるためにはもっと高速に Debian Women プロジェクトが成長する必要がある

B Debian Women という名前は性差別なので Debian people と呼ぶべきだ

C 女性のために簡単に入れる debian-women という新しいコミュニティをつくるのが目標ではなく、Debian Developer に参加するための援助をしていくのが目標なのでそれを忘れないように

問題 44. Andrews Schuldei が求めたのは

A Debian Developer たちが集まって作業するのと効率がよい場合もあるので、できる場所などの寄付

- B 個人的な利益
- C 世界人口の半分が Debian Developer になること

問題 45. Torsten Landschoff が提案したのは、共有ライブラリパッケージの SONAME が変わっただけのような場合に関しては自動で ftp-master が ACCEPT する仕組みだが、その仕組みに対しての反論は

A Joerg Jaspert 曰く、空のパッケージがつい最近アップロードされていたので、そういう間違いが検出できるのは重要だ。

B Joerg Jaspert 曰く、ftp-master の権限を弱めるような実装はありえない

C Joerg Jaspert 曰く、その実装の詳細な部分が気に入らない

問題 46. Gutenprint が unstable にリリースされた。以前はなんという名前だったか

A gimp-print

B gnome-print

C kde-print

## 9.8 2005 年 34 号

<http://www.debian.org/News/weekly/2005/34/> にある 2005 年 8 月 23 日版です。

問題 47. DCC に関して、Debian 商標についての決定権を委任されたのは

A Don Armstrong

B Andreas Schuldei

C Branden Robinson

問題 48. gotom さんが glibc について宣言したのは

A カーネル 2.4 しか動かないシステムではコンパイルできないぞ。

B ドアストップアーキテクチャについては、サポートはしない

C よくこわれるのでインストールしないでください。

問題 49. Steve Langasek によると、woody から sarge の間のための移行パッケージというのは

A 必要なので永遠に残す

B 途中のリリースを省略したアップグレードはサポートしないので、woody から sarge への移行専用のパッケージは unstable では削除すべき

C バグが多いので対応してほしい

問題 50. Ramakrishnan Muthukrishnan さんと Ganesan Rajagopal があつく Debian について語ったイベントは

A インドで実施した Debian Conference India

B 日本で開催した Debian Conference

C 中国で実施した Debian Asia Miniconf

問題 51. woody にしかない古いバグをクローズするのはどうしたらよいか

- A 「君の事はずっと嫌いだった」と書いたメールをそのバグに対しておくりつける
- B どのバージョンで問題が解決したのか、という情報を BTS に記録する
- C woody タグを付ける

問題 52. Lars が piuparts で実施してみたらどうなったか

- A たくさんバグが見つかった
- B Debian は完全であることが立証された
- C piuparts 自身が動かなかった

問題 53. バグ報告に GPG 鍵認証を要求する話についてどういう意見が出たか

- A Debian Developer 以外でもできるようにはすくなくともしてほしい
- B バグ報告は高尚な行為なので Debian Developer 以外は実施してはならない
- C GPG より違う認証プロトコルを使おう

問題 54. BTS の LDAP ゲートウェイはどうなったか。

- A master のポート 10101 にて稼働再開した
- B 止まったまま
- C 使えないのでステ

## 9.9 2005 年 35 号

<http://www.debian.org/News/weekly/2005/35/> にある 2005 年 8 月 30 日版です。

問題 55. Joerg Jaspart さんが発表した NEW キューで REJECT される要因は

- A DEB\_AUTO\_UPDATE\_DEBIAN\_CONTROL を使っている CDBS のパッケージは拒否
- B パッケージ名が glibc
- C メンテナ名が kmuto@debian.org

問題 56. Debian GNU/kFreeBSD では、Debian main の何\能か

- A 81.69%
- B 15%
- C 3%

問題 57. Andreas Barth さんがライブラリについて要求したのは

- A すでに移行が多数並行しすぎているため、ライブラリの SONAME の変更などは発生させないでほしい。
- B ライブラリは全部 static link に変更すること
- C ライブラリはすべてデバッグ情報を残す事

問題 58. バグ報告の影響をうけているユーザが投票することで、重要度をトラッキングするランキングを実施しようという意見に対して、現在すぐにでもできそうな方法は何か

- A バグに最後に活動した日付によってランキングを付ける
- B BTS と Popularity Contest を連携させる
- C 電波によって影響度を推察

問題 59. POSIX Shell として POSH を利用してスクリプトの試験を実施することに対しての反論は？

- A busybox のシェルですらそれよりも機能があるため、実質的な利益が無い
- B posh はつかえない
- C posh はバグだらけだ

問題 60. Mozilla についてセキュリティーフィックスとして新しい上流バージョンを利用することに関する理由は

- A めんどくさい
- B 上流がそれじゃないと嫌だというから
- C Ubuntu 向けに Martin Pitt がセキュリティーフィックスのバックポートをけっこうな時間をさいて実施してみたが、難しかった

問題 61. Yaroslav Halchenko が十分な AM の人数がいるのに 100 日以上かかっているのは NM プロセスが忍耐力を試験しているのではないかと、というコメントをした際の Marc Brockschmidt の回答は

- A 陰謀なので気づかないでください
- B 忍耐力を計測しているのにきついちゃった？
- C そんなことはなく、今一番活発な AM である Marc と Joerg は担当している NM の数を減らそうとしている最中なので、AM が増えるのは好ましい

問題 62. Joey hess が debconf に依存しているけど debconf-2.0 に依存していないパッケージをなくしたいのは

- A cdebconf に移行できるため
- B debconf2.0 の機能だけにしたいから
- C debconf のバージョンを新しくしたいから

問題 63. lists.debian.org にあるウェブ経由でのメールインタフェースでメーリングリストのスパムを報告するリンクは何をするものか

- A メーリングリストにスパムを投げる
- B メーリングリストにスパムが来てますというメールをメーリングリストの参加者にランダムに投げる
- C 実はまだ統計情報を集めているだけで何をするというものでもない

問題 64. ビルドするのに non-free なシステムが必要なパッケージは DFSG-Free か

- A DFSG-free でもビルドするのに non-free なものが必要なら DFSG-Free とは言えない
- B パッケージ自身のソースがフリーならフリーだろう
- C DFSG-Free の定義による

9.10 2005 年 36 号

<http://www.debian.org/News/weekly/2005/36/> にある 2005 年 9 月 6 日版です。

問題 65. KDE の C++ 移行の状況はどうなったか

- A kde と arts と qt が全部移行完了した
- B 永遠におわらなさそう
- C C++ってなんですか？

問題 66. Wiki の内容について何が議論になったか

- A wiki に書いてある内容のライセンスは何か
- B wiki の内容がみにくい
- C wiki つかいづらい

問題 67. curl に何がおきたか

- A openssl を使うようになった
- B gnutls 専用に移行した
- C ssl を独自で再実装した

問題 68. データベースを削除するか質問するのに debconf を利用することに関して joey hess は何をいったか

- A debconf が存在するのかをきっちり確認してなら、purge される際に debconf の質問をきいてデータを消すかどうか決めるのは悪いことではない
- B そんな重要な質問に debconf を使うな
- C いかなる場合でもユーザデータは削除するべきではない

問題 69. Marc Brockschmidt が要求したのは

- A パッケージをアップロードするさいにはちゃんと変更点を報告すること
- B API を変更しないでほしい
- C API が変更したときに通知すること

問題 70. Wolfgang Borgert が意味の無い README ファイルについて苦情を述べた。その例でなかったのは

- A 「Debian パッケージはあるので、apt-get install パッケージ名でインストールすることが可能です」と書いてある README ファイル
- B 「私の名前は中村です」と書いてある README ファイル
- C 「このファイルは XXX パッケージの README ファイルです」と書いてある README ファイル

9.11 2005 年 37 号

<http://www.debian.org/News/weekly/2005/37/> にある 2005 年 9 月 13 日版です。

問題 71. バグトラッキングシステムの見栄えで最近かわったのは何か

- A CSS を利用するようになった
- B DHTML になった
- C XHTML になった

問題 72. Debian UK で問題になったのは何か

- A メンバーが活動的でないこと
- B UK の経済状況がよろしくないこと
- C 商用利用をしようとした場合の Debian という名前の商標の利用の許可をする基準が不明確だったこと

問題 73. ソフトウェアを計測する、という論文で発表されたのは何か

- A Debian sarge には 2 億 3000 万行のソースコードが含まれている
- B Debian sarge の品質を計測した
- C Debian sarge の利用しやすさを計測した

問題 74. Joey Hess は testing に対して security 対応をすることを発表した。それに利用しているサーバはどれか

- A [secure-testing.debian.net](http://secure-testing.debian.net)
- B [security.debian.org](http://security.debian.org)
- C [security.debuan.org](http://security.debuan.org)

問題 75. `/usr/doc` をいまだにつかっているパッケージ数はどれくらいか

- A 100
- B 200
- C 500

問題 76. [planet.debian.org](http://planet.debian.org) をメーリングリスト経由で配布しようという意見に対して出た反論は

- A blog の内容は機密事項なので、メーリングリストで配布してほしくない
- B メーリングリストとして配布するとサーバの負荷が高くなる
- C blog の内容を永続的にメーリングリストのアーカイブとして保存されたくない

問題 77. `/usr/share/doc/パッケージ名/examples/` にあるファイルに実行権限をつけることについてはどうすべきか

- A サンプルは実行できるものは実行権限をつけるべき
- B サンプルなんてかざりなので実行しなくてよい
- C `/usr/share` 以下について実行権限をつけるのはこのましくなく、実行ファイルは `bin` におくべきだ。

問題 78. [sponsors.debian.net](http://sponsors.debian.net) が提供するサービスは何か

- A 金銭的寄付をつのるフィッシングサイト
- B 広告を配信し、広告収入を Debian プロジェクトの発展のために利用するサイト
- C まだメンテナになっていない人が管理しているパッケージについてスポンサーが必要な状況をトラッキングするシステム

問題 79. 1.0beta3 のようなベータ版のバージョン番号が 1.0 のような最終版のバージョン番号より低い、と dpkg が判定してしまう。この状況に関してメンテナはどう対応するべきか

- A 優先度の低いチルダ記号 ~ を利用して、1.0~beta3 のような名前にする。ただまだアーカイブシステムが対応していないので、今後の改善が必要。
- B あきらめる
- C ベータ版はパッケージ化しない

問題 80. ソースのみのパッケージのアップロードを可能にするという提案についての反論は何か

- A バイナリが必要でなくなると、メンテナがテストをしなくなるのではないだろうか、という懸念がある
- B ソースのみだとパッケージインフラが破綻する
- C katie を改変するのが面倒

問題 81. BTS に任意のタグを追加できる機能が追加された、なんという機能が

- A tagtag
- B たぐるんです
- C usertag

## 9.12 2005 年 38 号

<http://www.debian.org/News/weekly/2005/38/> にある 2005 年 9 月 20 日版です。

問題 82. David Moreno Garza が wnpp にて close したバグレポートの数は

- A 729 のバグレポート
- B 100 のバグレポート
- C 123 のバグレポート

問題 83. International Conference on Open Source Systems に投稿された論文の中で説明されていた結果は

- A 開発者は短期間でどんどん入れ替わる
- B メンテナは実は幻想で、そんな人は存在しない
- C 長いあいだアクティブに活動し、パッケージの数も多くメンテナンスする

問題 84. Frank Lichtenheld が発表したのは、non-free なドキュメントを削除する処理を開始するということだった。状況をトラッキングするために彼が利用したインフラは。

- A BTS の usertags 機能で debian-release@lists.debian.org ユーザのタグとして管理
- B Wiki ページ
- C CVS 管理のテキストファイル

問題 85. Software freedom day 05 で Debian-women が行って、結果として良かったので今後も継続することになったのは

- A debian-women-new IRC チャンネルがよい結果をもたらしたので、今後は debian-women チャンネルに新人を歓迎する時間帯というのをもうける
- B CD をたくさん焼いたら人気だった
- C Katie や BTSなどをインストールしてユーザがいじれるように提供したら人気だったので、今後もやる

問題 86. init.d スクリプトは現在直列に実行されているが、今後、並列実行を実装する際に便利だろうと思われる LSB 規格の仕様は

- A なんとなく並列に実行しても壊れないようにする仕様
- B 気持ちの中だけでは並列な年頃
- C init スクリプトの中で依存関係を記述できる仕様

問題 87. 新しいバージョンのパッケージにて問題が解決した場合の、バグレポートをクローズする方法でないのは何か

- A changelog でバグ番号を記述しアップロードする
- B バージョンヘッダを付けて、リクエストを -done アドレスに投げる
- C btsclose コマンドを利用する

問題 88. Marc Brockschmidt が説明した、新規メンテナプロセスの Front Desk の変更とは

- A 今後はより厳しい思想チェックを行う
- B Debian にコントリビュートしていることが要件になり、何もしていない場合は、応募が取り消される
- C 年齢制限を設けます

問題 89. security.debian.org で問題になったのは何か

- A セキュリティアップデートが遅い
- B セキュリティアップデートが嘘だった
- C xfree86 のセキュリティアップデートがあまりにも高いネットワーク負荷を発生させてしまい、security.debian.org がサーバとして機能しなくなってしまった。

## 9.13 2005 年 39 号

<http://www.debian.org/News/weekly/2005/39/> にある 2005 年 9 月 27 日版です。

問題 90. Ben Hutching が Debconf について報告したのは

- A もう終わってしまった事は忘れる
- B 忘れ物がありました
- C DVD が入手可能になった

問題 91. wiki.debian.org への移行で特に手動の労力が必要だったのはどこか

- A すでに wiki.debian.net から wiki.debian.org に移行してしまっているページがいくつかあったのでそれに対する手動の対処
- B kwiki から moinmoin ヘデータ形式の変更
- C ドメイン名の登録

問題 92. init の時点では/が read-only でマウントされているが、その時点でデータを保存するのはどうしたらよいか。

- A メモリファイルシステムを/run にマウントする
- B /mnt 以下にメモリファイルシステムをマウントする
- C /を rw にマウントしなおす

問題 93. グラフィックライブラリ GLU の実装が Debian 内で複数ある理由はなぜか

- A 一部のコードが一部のハードウェアでしか動かないという状況が続いているから
- B 複数のパッケージをメンテナンスしているほうがカッコいいから
- C メンテナの仲が悪いから

問題 94. Jeroen van Wolffelaar が提案したのは

- A libc5 を消す
- B libc6 を消す
- C libc6.1 を消す

問題 95. piuparts であきらかになる問題は

- A purge する際に、essential ではないパッケージに依存して、動作しないパッケージ
- B インストールしても動かないパッケージ
- C 使ってみて使いにくいパッケージ

## 9.14 2005 年 40 号

<http://www.debian.org/News/weekly/2005/40/> にある 2005 年 10 月 4 日版です。

問題 96. DPL チームが今後検討する予定の問題について記録する媒体として選択したのは

- A BTS
- B IRC bot
- C Wiki

問題 97. tetex 3.0 はどういう状況になっているか

- A 今後も入る見通しがない
- B うごかなくて困っている
- C experimental にアップロードされ、ライブラリのフリーズが完了したら unstable に入る

問題 98. Debian で配布する IA64 アーキテクチャ向けのカーネルについて Dann Frazier が SMP じゃないカーネルのサポートを削除しようとした、何故か

- A SMP じゃないといやだから
- B 時代は SMP です
- C IA64 で、SMP でないシステムがほとんどなく、あまりテストされていない

問題 99. Wolfgang Borgert によると planet.debian.org と、メーリングリストの利用方法の違いは

- A メーリングリストは古い技術なので今後はつかわなくなる
- B blog はフレームされないで意見を述べることのできるメディアだが、議論するのはメーリングリストでして欲しい
- C planet.debian.org は安定していないので使わないで欲しい

問題 100. pbuttonsd は /dev/input/eventXX を利用しているが、どういう問題があったか

- A makedev が、最大 32 あるうちの 4 個しかデバイスファイルをつくっていなかったため、/dev を静的に管理しているユーザは一部の機能を利用できていなかった。
- B USB 接続ではうまく認識できなかった
- C 電源ボタンがおされたらアプリケーションがハングした。

## 9.15 2005 年 41 号

<http://www.debian.org/News/weekly/2005/41/> にある 2005 年 10 月 11 日版です。

問題 101. Debian security で改善したのは

- A バックエンドとフロントエンドのサーバを分割し、負荷に強い構成に変更した
- B 特定のユーザが負荷をかけられないようにスロットリングした
- C セキュリティパッチをリリースしないことでサーバに負荷がかからないようにした

問題 102. Carlos Parra Camargo が報告したのは何か

- A Wiki が悪意をもったユーザにより書き換えられていたので前のバージョンを復活させた
- B Wiki がおもしろくないので改善しよう
- C Wiki サーバがダウンしている

問題 103. mozilla 1.7.8 に対してのセキュリティアップデートはどのような形でリリースされたか

- A 1.7.10 にバージョン 1.7.8 という名前をつけてリリースした
- B セキュリティパッチをバックポートした
- C セキュリティホールのある機能を全て disable にした

問題 104. 複数の chroot で同じユーザ情報を利用するのに利用できる方法でないのは

- A FUSE の shadow etc
- B LDAP
- C rm /etc/passwd

問題 105. ソースコードにローカルに適用したパッチをパッケージのアップグレード後も維持するためにはどうしたら一番楽か

- A 自分でがんばる
- B apt-src を利用する
- C パッチはあてない

問題 106. Jurij Smakov がリリースした文書は何か

- A Debian Users Handbook: Debian ユーザをどうあつかえばよいのか、が書いてある
- B Debian Developers Handbook: Debian Developer をどう扱えば良いか、が書いてある。
- C Debian Linux Kernel Handbook: Debian でカーネルがどうビルドされているのか、が書いてある

## 9.16 2005 年 42 号

<http://www.debian.org/News/weekly/2005/42/> にある 2005 年 10 月 18 日版です。

問題 107. Elive って何？

- A 電子的に生きること
- B enlightenment ベースの LiveCD
- C Elise の新しいバージョン

問題 108. m68k について Steve Langasek が発表したのは

- A m68k を自分もつかいたい
- B m68k が Debian の移植版の中で一番素晴らしい
- C testing に入る条件として、m68k は無視することにした

問題 109. etch 向けの debian-installer の状況はどうか

- A もう全アーキテクチャについてインストールできることは確認した
- B もうすでに完全に動いている
- C まだ一部のアーキテクチャではビルドできない

問題 110. gnome1 のパッケージがビルドできなくなったのはなぜか

- A 古いから
- B libpng10 が削除されたから
- C gnome2 の時代がやっときたから

問題 111. Edd Dumbill が、sarge をインストールする際に、debian-installer を利用している

ときにハードウェアの問題にあたったときに利用するように提案したのは

- A knoppix でハードウェア認識
- B ハードウェアを買い替える
- C Debian を使う事をあきらめる

問題 112. Oldenburg のミーティングの結果として、Debian security update で Branden Robinson が報告したのは

- A security.debian.org のバックエンドサーバが冗長構成になった
- B security.debian.org サーバが DNS のラウンドロビンで3台存在している構成がとれるようになった
- C security.debian.org がフィッシングサイトになった

問題 113. ソフトウェアに含まれている画像のライセンスに Creative Commons BY-SA ライセンスを利用したものは GPL のパッケージに含める事ができるか

- A やめたほうがよい
- B 可能
- C MJ Ray によると、不可能なため、そのような場合は MIT ライセンスを利用したほうがよい

問題 114. Camm McGuire が libbfd にリンクするにはどうしたらよいのだ、と質問したときの Daniel Jacobwitz の回答は

- A libbfd は安定しているのでいくらでもリンクしてくれ
- B よくバイナリレベルの互換性は破壊されるので、binutils-dev にある libbfd.a をつかってくれ
- C 一般人は libbfd は使わない

## 9.17 2005 年 43 号

<http://www.debian.org/News/weekly/2005/43/> にある 2005 年 10 月 25 日版です。

問題 115. Joerg Jaspert が NEW のパッケージを REJECT する理由で多いと指摘したのは

- A 読めないドキュメントが多い
- B おもしろくないパッケージが多い
- C debian/copyright が不正確なものが多い

問題 116. Steve Langasek が宣言した etch のリリーススケジュールによると、etch がリリースされるのは

- A 2005 年 12 月
- B 2006 年 6 月
- C 2006 年 12 月

問題 117. 今月、Christian Perrier が宣言したかなり完成している、とコメントしていた debian-installer の機能は何か

- A sid をインストールするインストーラ
- B etch 向けのテキストモードのインストーラ
- C GTK を利用したグラフィカルインストーラ

問題 118. ypbind などが動的にポートを確保する場合、その後に起動するサーバとポート番号がかぶる場合があるそれを回避する方法は

- A portreserve
- B 祈る
- C nis なんて使わない

問題 119. /etc/hosts に書いてある 127.0.0.1 のホスト名は現在の sid では何になるか

- A ホスト名
- B localhost.localdomain
- C localhost

問題 120. slang 用のモジュールパッケージ名は今後'slang-モジュール名' という形になりそうだが、従来はどういう名前だったか

- A sl モジュール名
- B モジュール名-slang
- C モジュール名

問題 121. pbuilder の開発体制にどのような変化があったか

- A 名前が変わりました
- B チームメンテナンス制をとるために、alioth に移動した
- C おもしろくなくなってきたのでもうやめます

問題 122. Daniel Ruoso が提案した Debian の移植版は

- A uclibc 移植版
- B minix 3.0 移植版
- C z80 移植版

## 9.18 2005 年 44 号

<http://www.debian.org/News/weekly/2005/44/> にある 2005 年 11 月 1 日版です。

問題 123. Nathanael Nerode が i386 のサポートについて何を宣言したか

- A そろそろ i386CPU も世界から駆逐できたので、無くす
- B gcc が i386 サポートを復活させたので、etch では真の i386 マシンでも動くリリースが作れるかも知れない
- C i386 向けのバイナリをとうとう生成できなくなってしまったので、今後は対応はしないことになる。

問題 124. Jay Berkenbilt が libtool の依存関係からパッケージの依存関係を計算するためのス

クリプトを作成したいという宣言を出した際に、問題になるだろうと指摘されたのは

- A multiarch の場合の .la ファイルが複数あるケースをどう扱うか
- B 思想的にそういうことはしてはならないことになっているので話題にあげるな
- C 自動で依存関係を解決するなんてそんな危険なことをしてはならない

問題 125. openssl の新しいバージョンがアップロードされたがその影響は

- A openssl に必要だったセキュリティー対応が取り込まれた
- B openssl がより安定したバージョンになった
- C 結果として 300 以上のパッケージをリビルドする必要がある

問題 126. berlinx では何が展示されていなかったか

- A nokia 770 で動いている Debian
- B Debian で制御している鉄道模型
- C Debian を使った炊飯器

問題 127. IETF の RFC について Simon Josefsson は何をしようとしているか

- A IETF から RFC を解放する運動を開始したい
- B RFC を利用しなくても世界がまわるようにあたらしい規格システムを提案、普及推進
- C RFC のライセンスをフリーソフトウェアに使いやすいように変更するための署名運動

問題 128. openafs-module-source の機構で問題になっているのは何か

- A インストールしても動かないため、それをごまかすための機構
- B ユーザに使い方を説明するためのマニュアルが大きすぎる
- C アップグレードする際にモジュールを自動で構築するように求める機構

問題 129. 自動テストの結果を上流に還元する方法について Daniel Jacobwitz は何を提案したか

- A パッケージのビルド中に結果を出力するようにする
- B 結果ログを上流に自動でメール送信するように設定する
- C 頑張っって手動で確認する

問題 130. Debian package で、postinst の処理がパッケージ毎に実行されてしまうために遅い場合がある。その処理を複数パッケージ分まとめて実行する方法にはなにがあるか

- A cygwin の setup.exe を利用する
- B そんな方法はない。
- C zope は apt の post-install を利用している。

問題 131. gnome メタパッケージが gnome-games に依存していることに出た苦情は

- A 政府機関ではゲームが禁止されているのでインストールされると困る。
- B gnome-games は大きすぎる
- C gnome-games はおもしろくないので、もっとおもしろいゲームを提供すべきだ。

9.19 2005 年 45 号

<http://www.debian.org/News/weekly/2005/45/> にある 2005 年 11 月 8 日版です。

問題 132. Linux-Info-Tag のブースではネットワーク障害があったが、どうやって対応したか

- A ノートパソコンの電源さえあればみんなハッピーだった
- B ノートパソコンに Debian のミラーがはいっていたので、ネットワークは特に問題ではなかった。
- C 通信衛星を利用してネットワークを仮稼働させたので問題にはならなかった

問題 133. Robert Milan が発表した ging は何をするものか？

- A CDROM から起動する Debian GNU/kFreeBSD
- B CDROM から起動する Windows
- C CDROM から起動する Hurd

問題 134. 次回の debconf の開催期間はいつか

- A 2005 年 5 月 14 日から 22 日
- B 2007 年 5 月 14 日から 22 日
- C 2006 年 5 月 14 日から 22 日

問題 135. openssl を利用している GPL のプログラムを gnutls を利用するようにするには

- A 気合いで書き直す
- B openssl を使い続けるほうが利点があるので、GPL のプログラムのライセンスを変えてしま
- う
- C gnutls/openssl.h という互換レイヤーがあるのでそれを利用すればよい

問題 136. <http://popcon.debian.org/> のトップページで見れない情報は何か

- A 最近各アーキテクチャにおいての popcon の利用者が急激に増えている
- B popcon のバージョンがリリースされた際にインストールされている数がどう遷移するか
- C どのパッケージが一番人気があるか

## 10 Debian Weekly News 問題回答

Debian Weekly News の問題回答です。あなたは何か問われましたか？

- |       |       |
|-------|-------|
| 1. A  | 39. C |
| 2. A  | 40. A |
| 3. A  | 41. C |
| 4. B  | 42. A |
| 5. B  | 43. C |
| 6. C  | 44. A |
| 7. A  | 45. A |
| 8. A  | 46. A |
| 9. A  | 47. A |
| 10. A | 48. A |
| 11. C | 49. B |
| 12. A | 50. A |
| 13. A | 51. B |
| 14. A | 52. A |
| 15. A | 53. A |
| 16. C | 54. A |
| 17. A | 55. A |
| 18. A | 56. A |
| 19. A | 57. A |
| 20. C | 58. B |
| 21. C | 59. A |
| 22. A | 60. C |
| 23. C | 61. C |
| 24. A | 62. A |
| 25. B | 63. C |
| 26. B | 64. A |
| 27. B | 65. A |
| 28. C | 66. A |
| 29. B | 67. A |
| 30. A | 68. A |
| 31. A | 69. C |
| 32. C | 70. A |
| 33. A | 71. A |
| 34. B | 72. C |
| 35. A | 73. A |
| 36. A | 74. A |
| 37. A | 75. C |
| 38. A | 76. C |
|       | 77. C |
|       | 78. C |
|       | 79. A |
|       | 80. A |

81. C  
82. A  
83. C  
84. A  
85. C  
86. C  
87. C  
88. B  
89. B  
90. C  
91. A  
92. A  
93. A  
94. A  
95. A  
96. C  
97. C  
98. C  
99. B  
100. A  
101. A  
102. A  
103. A  
104. A  
105. B  
106. C  
107. B  
108. C  
109. C  
110. B  
111. A  
112. B  
113. C  
114. B  
115. C  
116. C  
117. C  
118. A  
119. B  
120. A  
121. B  
122. A  
123. B  
124. A  
125. C  
126. C  
127. C  
128. C  
129. A  
130. C  
131. A  
132. B  
133. A  
134. C  
135. C  
136. C



あんどきゅめんてっど でびあん 2005 年冬号

2005 年 12 月 29 日 初版第 1 刷発行

東京エリア Debian 勉強会（編集・印刷・発行）

---